02/04/00

02-07-00

A

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
REQUEST FOR FILING NATIONAL PATENT APPLICATION
under 35 USC 111(a) and Rule 53(b) WITH SIGNED DECLARATION

PATENT APPLICATION

jc625 U.S. PTO 09/498525 02/04/00

Asst. Commissioner for Patents
BOX PATENT APPLICATION
Washington, D.C. 20231

NONPROVISIONAL

Sir:

Enclosed herewith is the PATENT APPLICATION of

Inventor: Hammad Elabd

Title: **Real Time DSP Load Management**

(Our Deposit Account No. 03-3975)

Our Order No.	73234	0261856
	Client #	Matter #
Atty. Docket	**73234**	**RCI-002**
	TMC#	Client Ref

including:

Date : February 4, 2000

1. [X] Specification: 39 pages 2. [ ] Specification in non-English

3. [X] Declaration [X] Original [ ] Facsimile/Copy

3(a) [X] Drawings: 25 sheet(s) [ ] informal [X] formal

4 [ ] AMEND the specification please by inserting before the first line: --This is a [ ] Continuation-in-Part [ ] Divisional [ ] Continuation [ ] Substitute Application (MPEP 201.09) of:

4(a) [ ] National Appln. No. filed --(M#)
4(b) [ ] International Appln. No. filed which designated the U.S.

5 [ ] See top first page re continuing application ("X" box only if info is there).

6 [X] An Assignment and cover sheet. Please return the recorded Assignment to the undersigned.

7 [ ] Prior application is assigned to by Assignment recorded , at Reel/Frame:

8 [ ] **FOREIGN** priority is claimed under 35 USC 119(a)-(d)/365(b) based on filing in

9.

Application No.	Filing Date	Application No.	Filing Date
(1)		(2)	

10. _ Certified copy/ies [ ] enclosed [ ] previously filed on in U.S. Application No. filed on

11 [X] Enclosed: (#) Verified Statement/s establishing "small entity" status under Rules 9 & 27.

12 [ ] DOMESTIC/INTERNATIONAL priority is claimed under 35 USC 119(e)/120/365(c) based on the following provisional, nonprovisional and/or PCT international application(s):

Application No.	Filing Date	Application No.	Filing Date
(1)		(2)	

13. [ ] This application is filed under Rule 53(b)(2) since an inventor is named in the enclosed Declaration who was not named in the prior application

14 [ ] Preliminary Amendment:

THE FOLLOWING FILING FEE IS BASED ON CLAIMS AS FILED LESS ANY ABOVE CANCELLED

15. Basic Filing Fee				$690 / 345	$ 345.00	101/201
16. Total Claims:	32	minus 20 =	12	x $18/$9 =	+ 108.00	103/203
17. Independent Claims:	5	minus 3 =	2	x $78/$39 =	+ 78.00	102/202
18. If multiple dependent claim is present, add				+ $260/$130	+	104/204
19. **TOTAL FILING FEE ENCLOSED =**					**$ 531.00**	
20. If "non-English" box is X'd, add Rule 17(k) processing fee				+ $130/$130	+	139
21. If "assignment" box is X'd, add recording fee				+ $40/$40	+ 40.00	581
22. [ ] Enclosed is a Petition/Fee under Rule No.				+ $130/$130	+	122
23. TOTAL FEE ENCLOSED:					**$ 571.00**	

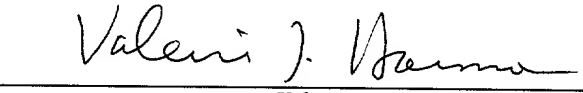
CHARGE STATEMENT: The Commissioner is hereby authorized to charge any fee specifically authorized hereafter, or any missing or insufficient fees which may be required under Rules 16-18 (missing or insufficient fee only) now or hereafter relative to this application and the resulting Official document under Rule 20, or credit any overpayment, to our Account/Order Nos. shown in the heading hereof for which purpose a duplicate copy of this sheet is enclosed. This CHARGE STATEMENT does not authorize charge of the issue fee until/unless an issue fee transmittal form is filed.

1100 New York Avenue, N.W.
Ninth Floor, East Tower
Washington, D.C. 20005-3918
Tel: (650) 233-4776
Fax: (650) 233-4545

PILLSBURY MADISON & SUTRO LLP
By: **Chang H. Kim**, Reg. No. 42,727

Express Mail Label: EL 513850778 US
Date of Deposit: February 4, 2000

I certify that this paper and listed enclosures are being deposited with the U.S. Post Office "Express Mail Post Office to Addressee" under 35 CFR 1.10 on the above date, addressed to Asst. Commissioner for Patents, Box Patent Application, Washington, D.C. 20231



Valerie J. Harmon

Inventors: Hammam Elabd
App. No.: Unassigned
Filed: Herewith

Atty Dkt. 73234 / 0261856
Client Ref: RC-002

Title: **Real Time DSP Load Management System**

VERIFIED STATEMENT (DECLARATION) CLAIMING SMALL ENTITY STATUS (37 CFR 1.9(d) and 1.27(c)) - SMALL BUSINESS CONCERN

I hereby declare that I am an official empowered to act on behalf of the small business concern identified below:

NAME OF CONCERN: **REALCHIP INC.**
ADDRESS OF CONCERN **1290 Oakmead Parkway, Suite 318, Sunnyvale, CA 94086**

I hereby declare that the above identified small business concern qualifies as a small business concern as defined in 13 CFR 121.12, and reproduced in 37 CFR 1.9(d), for purposes of paying reduced fees under Section 41(a) and (b) of Title 35, United States Code, in that the number of employees of the concern, including those of its affiliates, does not exceed 500 persons. For purposes of this statement, (1) the number of employees of the business concern is the average over the previous fiscal year of the concern of the persons employed on a full-time, part-time or temporary basis during each of the pay periods of the fiscal year, and (2) concerns are affiliates of each other when either, directly or indirectly, one concern controls or has the power to control the other, or a third party or parties controls or has the power to control both.

I hereby declare that rights under contract or law have been conveyed to and remain with the small business concern identified above with regard to the invention entitled as above and invented by **ELABD** described in the above-captioned specification.

If the rights held by the above-identified small business concern are not exclusive, each small entity, individual, concern or organization having rights to the invention is listed below* and no rights to the invention are held by any person, other than the inventor, who could not qualify under 37 CFR 1.9(c) as an independent inventor if that person had made the invention, or by any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e). *Note: Separate verified statements are required from each person, concern or organization having rights to the invention, averring to small entity status (37 CFR 1.27).

FULL NAME of _____

[ ] INDIVIDUAL [ ] SMALL BUSINESS CONCERN [ ] NONPROFIT ORGANIZATION

I acknowledge the duty to file, in this case, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate. (37 CFR 1.28(b))

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

NAME OF SIGNATORY: Craig Slayter
TITLE: President
ADDRESS: 1290 Oakmead Parkway, Suite 318
Sunnyvale, CA 94086

SIGNATURE _____ DATE 2/3/2000

REAL TIME DSP LOAD MANAGEMENT SYSTEM

FIELD OF THE INVENTION

The present invention relates to a DSP load management system, and more particularly, to a system and method for enhancing the processing capabilities of a system on chip (SOC) device. The DSP load management system of the present invention enables parallel processing of data at high frequency. The present invention is further directed to a system and method for distributing, reading and writing data to several CPUs and/or DSPs simultaneously or in the same clock cycle. In addition, the DSP load management system provides forward-looking real-time evaluation of arriving data and an algorithm to optimizing loading for multiple DSPs before, during, and after algorithm switching operations.

BACKGROUND OF THE INVENTION

An SOC device has many advantages and benefits over a separate component integrated circuit (IC) device. For example, the separate IC device generally includes components that are connected to each other on a printed circuit board, whereas the SOC device includes an entire system (processors, memory, logic, clock, I/O control unit, etc.) embedded on a single chip, thereby producing a device that is smaller, faster, and more efficient. Furthermore, the overall size of the end product is reduced using the SOC device because manufacturers can put major system functions on a single chip, as

opposed to putting them on multiple chips. As is well known, the SOC device has at least an embedded processor (e.g., ARM, LEXRA, MIPS, ARC, DSP core), memory, and logic.

The SOC device also provides faster chip speeds due to the integration of the components/functions into a single chip. Many applications such as high-speed communication products (VoIP, MoIP, wireless, imaging) require chip speeds that may be unattainable with separate IC components. This is primarily due to the physical limitations of moving data from one chip to another, through bonding pads, wires, buses, etc. Integrating components/functions into one chip eliminates the need to physically move data from one chip to another, thereby producing faster chip speeds. Further, the SOC device consumes less power than the separate IC device since data do not need to be moved from one chip to another. Another advantage of using the SOC device is that it is less expensive for the manufacturer because of the reduced number of chips used in the end product. Packaging costs, which can be significant, are likewise reduced as a result of having fewer chips. Thus, SOC devices are becoming ever more popular and are widely used in many applications requiring fast/high data rates as in the Internet/imaging applications.

However, one major shortcoming associated with the SOC device is that there may be multiple DSPs (digital signal processor) on a single chip, which requires one or more CPUs to process data at a very high rate. As a consequence, the one or more CPUs may not be able to efficiently perform general system/application tasks since each request from the DSPs consumes tens or hundreds of CPU clock cycles.

Typically, a 16 bit DSP array can handle two DSP words, 16 bit each at/during

one clock cycle during operation and, thus, corresponding memories such as SDRAMs will likewise have 16 bit wide buses. It is also known that on a single board, performance speeds of greater than 100 MHz is quite difficult to achieve for DSPs with 16 bit capabilities. As a result, a 100 MHz bus is generally implemented between the DSPs and memories.

To further illustrate the shortcoming of the prior art system, assume that a burst operation is performed, which requires 12 cycles (120 nanoseconds) to read 8 x 16-bit words. Assume also that the capability of each memory access controller (MAC) is about 133 MB/sec under the most ideal condition during the burst operation, without reading a single word. As known, the CPU other operations (system, periphery devices, etc.) generally require at least 80 MB/sec or more. This means that the CPU cannot perform all the functions for the multiple DSPs and MACs. For example, it is known that there are no less than 60 MB/sec for DSP operations. Thus, multiple threads between the CPU and the DSPs are needed to operate an efficient overall system.

In general, the number of DSPs required in an application is a function of the input sample rate, algorithm MIPS (million instructions per second), number of different algorithms employed, memory access overheads (i.e., arbitration, latency), I/O data distribution overheads (input/output data), and time skew. Thus, the total throughput on the DSPs is a function of their workloads and overheads. Further, the workload distribution is a function of the present and new loads along with the present and new algorithms that may be implemented in the future.

One design option for implementing the SOC device with multiple DSPs is to provide individual embedded memory for each DSP. This method, however, requires a

great deal of memory space and is often very costly and inefficient when implemented using a conventional CMOS logic process. Alternatively, it can be implemented using an embedded DRAM process, but this process is also very costly. One advantage of the SOC approach is that circuitry that is on a board can be implemented on a single chip, thereby reducing AC power dissipation of memory access controllers as a result of integrating more memory on the chip.

As is well known, an embeddable DSP is presently operational in the range of 125 - 180 MHz. However, there are many systems that are currently being designed that are operational in the 800 MHz to 2 GOPS/s range. Embeddable DSPs with 800 MHz to 2 GOPPS/s clocks are currently not available for such systems. To provide the functional equivalence for the above system, the first design option is to design a system with multiple DSPs connected in a network (i.e., seal gate for image processing). The second design option is to use combination of multiple programmable DSPs and processing element DSP array networked together by a DSP load management system for efficient implementation of multiple DSP algorithms. The present invention is implemented using the second design option.

Fig. 1 illustrates a simplified block diagram of a conventional SOC device having a CPU 2 and multiple DSPs 12a, 12b, 12c. Two separate chips are required in this device, one for the CPU 2, and one for the DSPs 12a, 12b, 12c and their program and data memories (shown in Fig. 2). Only one CPU 2 and three DSPs 12a, 12b, 12c are illustrated herein, but it is understood that more than one CPU and more or less than three DSPs, or other kinds of processors can be used. The CPU 2 includes a cache 4, which may be internal or external, as known in the art. Other conventional components, which

are not illustrated herein, may also be connected/coupled to the CPU 2 and DSPs 12a, 12b, 12c.

Also included in the diagram of Fig. 1 is data1 20a, data2 20b, and data3 20c, which data is associated with an outside communication port. Data1 20a, data2 20b, and data3 20c can be data originating from a telephone (voice), fax, modem or combinations thereof. When the CPU 2 fetches the data from the communication port, data from data1 20a, data2 20b, and data3 20c are broken up and buffered in the CPU cache 4. The buffered data 6a, 6b, ...6n, is transmitted to the appropriate DSP 12a, 12b, 12c, based on intelligent software decisions made by the CPU 2. For example, during each clock cycle, a particular data (e.g., 6a) originating from data1 20a in the cache 4 can be sent to DSP 12a, and data 6b originating from data2 20b in the cache 4 can be sent to DSP 12c. Transmitting a predetermined number of data from the cache 4 to the DSPs 12a, 12b, 12c, works well as long as the CPU 2 does not have to perform other functions and the data is slow (i.e., 1 or 2 mega-bits source). However, this schema will fail if the data is fast (i.e., imaging, fast communication ports such as T3 45 Mbps, etc.) and when the CPU 2 has to perform other application or system functions or when arbitration the memory reduces data throughput.

Fig. 2 illustrates a block diagram illustrating the SOC device of Fig. 1 having the CPU 2 and multiple DSPs 12a, 12b, 12c, where the CPU 2 is transmitting data to the multiple DSPs 12a, 12b, 12c. As shown, DSP1 12a communicates with DSP program and data memory 22a, which memory is designated for only DSP1 12a. Likewise, DSP2 12b communicates with its designated program and data memory 22b, and DSP3 12c communicates with its designated program and data memory 22c. In this example, DSP1

12a, DSP2 12b, and DSP3 12c are assumed to have program and data memory and tables in their memories 22a, 22b, 22c. When data buffered in the CPU cache 4 corresponds to algorithm 3 running in the DSP3 12c, then data is sent to the DSP3 12c for processing since the program and data memory 22c includes a program for algorithm 3. On the other hand, when data buffered in the CPU cache 4 corresponds to algorithm 1, 2, it is sent to either/both DSP1 12a or DSP2 12b since program and data memories 22a, 22b include algorithms 1, 2. After the data is processed by the DSPs 12a, 12b, 12c, the processed data can be transmitted to an external source.

When the contents of the program and data memory needs to be switched, one or more of the program and data memories 22a, 22b, 22c will be flushed or cleared out so that new memory can be loaded therein. New data and/or program tables are retrieved from external memory and loaded into one or more memories 22a, 22b, 22c. As illustrated, each DSP1 12a, DSP2 12b, DSP3 12c also communicates with its corresponding external memory. This process is a major undertaking since it is very time consuming, as the CPU 2 attempts to read from external memory and load DSP program and data tables in the program and data memories 22a, 22b, 22c.

As can be appreciated, the conventional system is very limiting since switching algorithms can be time-consuming. In addition, the conventional system is designed to only handle data with very slow rates. Accordingly, there is a need for a DSP load management system (DLMS) that works simultaneously in conjunction with the CPU(s), DSPs, memory, and memory management system to provide efficient switching and handling of data with high rates and efficient loading or switching of algorithms and data tables into DSP memories.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a DSP load management system that enhances the processing capabilities of an SOC device.

It is another object of the present invention to provide a DSP load management system that enables parallel processing of data at high frequency from shared memory.

It is still another object of the present invention to provide a DSP load management system that distributes, reads and writes data to several CPUs and/or DSPs in the same clock cycle.

It is another object of the present invention to provide a DSP load management system that includes information relating to the internal characteristics of an DSP array.

It is a further object of the present invention to provide a DSP load management system that requests multiple bursts from a memory access controller via multi-threaded memories.

It is another object of the present invention to provide a DSP load management system that diverts tasks from one DSP to another, with short or zero latency.

It is still a further object of the present invention to provide a DSP load management system that optimizes dynamic algorithm switching in real time.

It is another object of the present invention to provide a DSP load management system that loads algorithms and data tables in DSP memories in real time.

It is yet another object of the present invention to provide an intelligent DSP load management system that allows the CPU to perform system control and application tasks without the CPU having to perform word by word data loading and instruction by

instruction switching functions.

It is another object of the present invention to provide a system and method for dynamically loading data in multiple DSPs based on real-time processing capabilities of the multiple DSPs.

It is a further object of the present invention to provide a system and method of representing a DSP array as a single DSP system with a wide bus.

It is yet another object of the present invention to provide a system and method for threading a data stream from one processor to another.

It is yet another object of the present invention to provide a system and method having hardware and software functionalities for managing the load of an DSP array.

It is a further object of the present invention to provide programmable DSPs and networking them together for efficiency.

It is another object of the present invention to provide a system and method having hardware and software functionalities for optimizing IP packet delay and for increasing quality of service based on the type of service and priority.

It is yet another object of the present invention to provide a system and method for allowing the DSP load management system to interface between one or more CPUs and one or more DSPs.

It is yet a further object of the present invention to provide a system and method providing forward looking real time evaluation of arriving data.

These and other objects of the present invention are obtained by providing a DSP load management system that enhances the processing capabilities of an SOC device. The DSP load management system can be interfaced between one or more CPUs, one or

more DSPs, and/or a memory management system or memory access controllers for enabling parallel processing of data at high frequency and loading and switching DSP programs without causing arbitration conflicts on the memory access bus. Data can be distributed, and read/write to several CPUs and/or DSPs in the same clock cycle. In addition, the DSP load management system provides forward-looking real-time evaluation of arriving data. The present invention also provides a system and method for distributing and re-distributing loads between several DSPs.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects and advantages of the present invention will become apparent and more readily appreciated from the following detailed description of the presently preferred exemplary embodiments of the invention taken in conjunction with the accompanying drawings, of which:

Fig. 1 illustrates a block diagram of a conventional SOC device having a CPU and several DSPs;

Fig. 2 illustrates a block diagram illustrating a SOC device of Fig. 1 having a CPU transmitting data to several DSPs;

Fig. 3A illustrates a block diagram of an SOC device having a DSP load management system in accordance with the preferred embodiment of the present invention;

Fig. 3B illustrates a detailed block diagram of an SOC device having a DSP load management system in accordance with the preferred embodiment of the present

invention;

Fig. 3C illustrates a block diagram of a specific implementation of a DSP load management system in accordance with the preferred embodiment of the present invention;

Figs. 4A-4F illustrate the intra-transaction load optimization process using the DLMS in accordance with the preferred embodiment of the present invention;

Figs. 4G-4H illustrate the intra-transaction DSP algorithm switch optimization in accordance with the preferred embodiment of the present invention;

Figs. 5A-5C illustrate an example of the DSP MIPS budget using three DSPs and two algorithms in accordance with the preferred embodiment of the present invention;

Figs. 6A-6C illustrate an example of the DSP MIPS budget using three DSPs and three algorithms in accordance with the preferred embodiment of the present invention;

Figs. 7A-7B illustrate a schematic diagram of a coprocessor interface, CPU, and DSP in accordance with the preferred embodiment of the present invention;

Figs. 8A-8B illustrate multi-coprocessor interfaces that can be implemented in accordance with the preferred embodiment of the present invention;

Fig. 9A illustrates a system including a CPU, CI, DLMS, MT-MMS and multiple DSPs in accordance with the preferred embodiment of the present invention;

Fig. 9B illustrates an effective behavior model of the system in Fig. 9A in accordance with the preferred embodiment of the present invention;

Fig. 10 illustrates a MT MMS loading DSP with high frequency data in accordance with the preferred embodiment of the present invention;

Fig. 11 is an example of a two threaded memory which is used with two memory

access controllers in accordance with the preferred embodiment of the present invention;

Fig. 12 illustrates a system including a CPU connected to memory access controllers in accordance with the preferred embodiment of the present invention;

Fig. 13 illustrates a specific implementation of DMA and MAC cluster in accordance with the preferred embodiment of the present invention;

Fig. 14 illustrates a diagram of multiplexing the data bus to individual channels in accordance with the preferred embodiment of the present invention;

Fig. 15 illustrates a flow chart implementing the real time load management system in accordance with the preferred embodiment of the present invention;

Fig. 16 illustrates a block diagram of the DSP load management system parallel word transfer in accordance with the preferred embodiment of the present invention;

Fig. 17 illustrates a block diagram of the PE array controller in accordance with the preferred embodiment of the present invention; and

Figs. 18A-18C illustrate a system having a configurable bridge/gateway for optical networks in accordance with the preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention will be described in greater detail, which may serve to further the understanding of the preferred embodiments of the present invention. As described elsewhere herein, various refinements and substitutions of the various embodiments are possible based on the principles and teachings herein.

The preferred embodiments of the present invention will now be described with reference to Figs. 3-18, wherein like components and steps are designated by like reference numerals throughout the various figures. Further, specific parameters such as algorithms, loads, speeds, data rates, and the like are provided herein, and are intended to be explanatory rather than limiting.

The present invention relates to a system and method that combines hardware and software functionalities to provide a DSP load management system (DLMS). The present invention allows a single chip to include one or more CPUs, a DLMS layer, and multiple DSPs. Each DSP can access an external memory and an internal memory via its designated thread, as described in more detail below. The DLMS converts low power processing DSPs to high power processing DSPs. One or more CPUs can look up status tables/registers in the DLMS to obtain real time status of the multiple DSPs. This allows the CPU to determine which DSP is loading/processing what load and/or algorithm at any give time/cycle.

The present invention also provides inter-transaction, intra-transaction, and hardware mapping optimization schemes. In inter-transaction optimization, new calls are given to the least loaded DSP on frame boundaries, while in intra-transaction optimization, 2 or more algorithms can be switched on the same DSP during transaction time. In the hardware mapping optimization, same calls/samples are threaded between several DSPs during one frame time. These optimization schemes are discussed in greater detail hereinafter.

In general, the DLMS of the present invention can include the following functional components, which are described in greater detail later herein: (1) multiple

direct memory access (DMA) to memory system with internal DMA registers; (2) interface to the CPU (coprocessor interface - CI); (3) interface to DSP array; (4) interface to multi-threaded memory management system; (4) interface to DSP internal data and control registers; (5) interface to internal DSP program/data memory and ping-pong memory; (6) status and control registers; (7) FIFOs and state machines (SM); (8) MUXs and registers for simultaneous parallel transfer of data from one DSP to another; and (9) CPU instruction decoder to decode custom CPU instructions and generate multiple threading operations from the CPU instructions.

The DLMS also provides buffer memory for data transfers between multiple processors (CPUs, DSPs, etc) (i.e., registers to read out data from one processor and write to another processor). This can be accomplished by providing parallel word exchange between one or more DSPs and between one or more DSPs and memory (multi-threaded exchange).

In addition, the DLMS includes a decoder that is capable of decoding CPU instructions that require parallel data word transfers between DSPs and between DSPs and memories. The CPU instructions can be as simple as writing a bit to a control register or as complex as decoding a real-time instruction operation code. The DLMS can be configured to enhance instructions as well as data parallelism in the parallel DSP system. To enhance instruction parallelism, the DLMS transfers certain operations to the parallel processor array, e.g. 100 small processing elements, having identical or different processing elements that can be implemented with only a few instructions. In this manner, the DLMS and the CPU will need to recognize sections of the algorithms that are similar in nature so that they can be processed in parallel. Further, the DLMS control

registers are capable of reconfiguring the parallel function of the DSP array, i.e., via mode register.

The DLMS can also be considered as a load manager for a communication SOC that is used for complex processing of multi-channel, multi-service systems. The communication SOC can handle multi-channel, multi-protocol, and multi-service voice/fax/data with multiple digital signal conditioning/modulation (DSP) algorithms and multiple packet and cell processing protocols using multi-channel voice compression and multi-protocol packet and cell processing. The communication SOC integrates application-specific software such as TCP/IP protocol stacks or ATM SAR (segmentation and re-assembly) functions that are optimized for 16-and 32-bit data-streaming applications.

When the DLMS functions as the load manager, it divides the load between DSP cores such as Carmel DSP from Infinion and individual SOC processing elements such as a multiply and accumulate. As known, 16x16 or 32x32 bit MACs are usually available on the SOC as a part of the CPU core. These MACs can then be used as a substitute for an additional DSP core in algorithm implementation, where the interface between the DSP cores and the CPU-MAC includes the DLMS registers.

Fig. 3A illustrates a block diagram of an SOC device having a DSP load management system (DLMS) in accordance with the preferred embodiment of the present invention. The DLMS described herein is an intelligent device on the SOC device that manages the DSPs, CPUs, other processors, and memory management systems.

The DLMS 100 is interfaced between one or more CPUs 2a, 2b, and one or more DSPs 12a, 12b, 12c, 12n. The DLMS 100 also communicates with a multi-tasking

memory management system (MT MMS) or memory access controllers (MAC) 30 that is controlled by one or more CPUs, as described in the co-pending U.S. Application Serial No. 09/443,934, entitled "Multi-Tasking Memory Management System", the contents of which are expressly incorporated herein by reference.

The DLMS 100 can be disabled (made non-active) via software commands, which allows the CPUs 2a, 2b to communicate with the DSPs 12a, 12b, 12c, 12n as in the conventional manner. However, the DLMS 100 can be activated in order to allow the overall system to become more efficient. The DLMS 100 can (1) process on a transaction boundary basis or (2) optimize loads within a transaction, as described in more detail hereinafter. The DLMS 100 can also be programmed by one or more CPUs 2a, 2b via control registers to access memory through the MT MMS 30 using multiple threads. Each DSP 12a, 12b, 12c, 12n, includes a program/data memory and control/data registers.

Fig. 3B illustrates a detailed block diagram of an SOC device having a DLMS in accordance with the preferred embodiment of the present invention. Each of the DSP1, DSP2, DSP3, DSP4 in the DSP array 12 has access to its own separate or semi-separate external memory Ext MEM1, Ext MEM2, Ext MEM3, and Ext MEM4, respectively. There are also program and data and "ping-pong" memories 36a, 36b, 36c, 36d for the DSP1, DSP2, DSP3, and DSP4, respectively. The ping-pong memories are essentially on-chip buffer memory filled by DMA from an external memory. This type of architecture prevents multiple DSPs from competing with each other for access to the memory bus by way of providing multi-treaded memories. For example, the CPU 2 can essentially write to one section of the internal memories 36a, 36b, 36c, 36d (i.e., SRAMs having 1-6 transistors each) in the program/data memories, while another section of the

internal memories can be essentially read by the DSPs via the ping-pong memories.

In greater detail, the DLMS 100 includes multiple registers 32a, 32b, 32c, 32d, and multiple DMAs 34a, 34b, 34c, 34d, which include state machines (SM) (i.e., logic) and FIFOs (typically 32 bits). During operation, the CPU 2 using appropriate software can write in the DLMS registers 32a, 32b, 32c, 32d the beginning address and length of data, which the DMAs 34a, 34b, 34c, 34d need to retrieve. The DMAs 34a, 34b, 34c, 34d then establish a request via the MT MMS or MACs 30 to read the beginning address and length of data. There can be one MT MMS or four memory access controllers 30, as shown. The DMAs 34a, 34b, 34c, 34d can also provide read/write bursts to both internal and external memories via the bus 35 on the chip and can read data/instructions from the CPU 2 cache.

The state machine (SM) can initiate the transfer of data between the FIFO and the ping-pong memory in the DSP subsystem. In this manner, the FIFO can be filled and emptied continuously in a simultaneous manner.

During operation, a communication device sends communication data to the CPU 2. The CPU 2 then writes the data to the appropriate register in the DLMS 100 depending on which DSP is to perform the task. Status tables in the DLMS 100 can be used by the CPU 2 to keep track of which DSP is performing/processing what task and algorithm. At any point during the frame, the DLMS 100 can inform the CPU of the status. Instead of the CPU 2 doing all the processing, the DLMS 100 creates a status table of all the DSPs' conditions such as "finished processing", loading, etc. This assists the CPU 2 using, for example, a routing MUX 40 to determine into which register the next set of data should be written. While the DSPs are processing the data in the

background, the CPU 2 and the DLMS 100 are preparing the processing of the next data/algorithm. The routing MUX 40 is also coupled to the MT MMS or MACs 30, and may be coupled to a second CPU 2p.

The CPU 2 can look up the status table in the DLMS 100 to determine where to send the next set of data or the routing MUX can perform the task for the CPU 2. Each piece of data in the register is preferably a TDM (time division multiplex) call or PCM (pulse coded modulated) call. After processing the data using the DSPs, the data is preferably compressed to Internet packets and transmitted back to the CPU and then to an IP communication device.

The DMAs 34a, 34b, 34c, 34d in conjunction with the memory access controllers/MT MMS 30 and the CPU 2 transfers data between external memories and the FIFOs in DMA. The arbitration for the memory bus is done by an arbiter in the MAC or MT MMS 30. The DLMS 100 preferably includes at least the same number of DMAs 34a, 34b, 34c, 34d as the number of processors, DSPs. The selection of control status registers (CSR) for each DMA 34a, 34b, 34c, 34d is done by address decoding. The address range for each block is hardwired in the CSR register address decoder. The DLMS 100 accepts data either from the CPU 2, which writes into the DSP general or control registers or from the MT MMS 30 by transferring the data from the various DMA FIFOs to the ping pong program and data memories.

Each DMA FIFO has a finite size of, for example, 32 words. When it is filled below a minimum threshold value, a request is sent to the memory bus for more data. On the other hand, when it is filled above a certain maximum threshold value to avoid FIFO overflow, it will terminate data transfer from the memory to it. There are both read and

write FIFOs for each DMA interface.

Memory to DMA FIFO can be initiated by writing a non-zero value into data available (ready to read) register. The address from which data has to be transferred is specified by writing into the DMA address register. DMA to DSP program/data memory transfer (i.e., from DMA FIFO to DSP memory) is performed by the DMA signaling to the DLMS 100 that there is a new algorithm available by setting the send request signal. The DLMS 100 will then generate strobe signal to read the data from the DMA FIFO to the DSP program or data memory. The DLMS 100 also will generate the appropriate memory address to store the new algorithm in the DSP program memory and generate an interrupt signal once all the FIFO contents are sent to the DSP. The process will be repeated until all the new algorithm is written into the DSP program memory. At the end of the present data frame, the DLMS 100 will send a signal to the DSP to switch from the first field to the second of the program/data memory and ping-pong memory 36a, 36b, 36c, 36d.

In further detail, the CPU 2 can receive high frequency serial data from one or more communication cores/systems such as VoIP (voice over IP), FoP (fax over IP), IP to sonet, Read-Solomon, encoding, decoding, satellite receiver, broadband communication, image processing, data modem, and the like. The CPU 2 then compiles this data into frames of, for example, frame 1 covering the period between 0 to 10 msec, frame 2 covering the period between 10 to 20 msec, etc. These frames are then sent one at a time to one or more DSPs in the DSP array 12 using the DLMS 100. The DLMS 100 cooperates with the CPU 2 to distribute sections of the frames to different DSPs based on a load management algorithm programmed in the CPU 2 and the DLMS state machines.

Between each frame period or during the frame processing period, the DLMS 100 receives a change of frame early warning signal from the CPU 2 based on new data. The CPU 2 then sends a beginning memory address and block length information to the DLMS 100, which information is written into its DMA registers. The DMA then requests to several MACs or MT MMS 30 to retrieve new algorithms from external memory, if needed. The data and algorithms are then written into the DMA FIFO and then transferred by the DLMS state machines into the ping-pong memory via the DSP program/data memory. The data and programs are loaded in the same clock cycle into multiple memories or registers. The DMA burst transactions continue until all the algorithm/data is read. The CPU 2 can also write to control registers of the DLMS 100 to achieve synchronous or parallel DMA operations while reading the status registers, thereby saving itself instruction cycles.

The advantage of this invention is that external memory is used instead of internal memory on the chip. Manufacturing costs can be reduced as a result since embedded DRAM is not used. This invention also allows data to be threaded between DSPs at frame boundaries. Thus, several DSPs look like a very large and powerful DSP array with a very wide memory bus.

Fig. 3C illustrates a block diagram of a specific implementation of a DLMS in accordance with the preferred embodiment of the present invention. It is important to note that other system architectures and components can be used in the present invention than those described herein.

Figs. 4A-4F illustrate the intra-transaction load optimization process using the DLMS of the present invention. Figs. 4A-4F illustrate tables corresponding to four DSPs

having the identical maximum load capacities (Max MIPS) and same beginning algorithms. It is assumed that DSP1 currently has a load L1, DSP2 has a load L2, DSP3 has a load L3, and DSP4 has a load L4. In Fig. 4B, when an additional load L5 (assume to have same algorithm) is introduced, the DLMS will recognize that load L5 has the same algorithm as loads L1-L4, and load it into the least loaded DSP at that time. In this case, load L5 is loaded into DSP 3 since it is the least loaded DSP among the four DSPs. Load L5 may be transmitted from a CPU cache or from memory threads using the MT MMS.

Next, assume that an additional load L6 with a different algorithm is required to be loaded in one of the four DSPs. In this case, one of the DSPs, i.e. DSP1, will be cleared or flushed out so that the load L6 can be loaded therein. DSP1 is selected for loading load L6 because DSP1 is the least loaded DSPs, or alternatively, because of other pre-programmed instructions. In this example, the DLMS switches the load L1 from the DSP1 into DSP2 and DSP3 since DSP2 and DSP3 are least loaded than DSP4. In other examples, load L1 may be loaded in only one rather than two DSPs. After the load L1 is cleared or flushed out of DSP1, the load L6 is loaded therein as illustrated in Fig. 4C. Before load L6 can be loaded, the entire contents of the program memory and data tables need to be switched in DSP1. This is accomplished by the CPU writing to the DLMS registers to achieve direct memory access of program code in burst mode, as discussed in more detail later herein. The DSP1 pre-fetches the new program memory and data tables before the load L6 is loaded therein.

When an additional load L7 with the same algorithm as load L6 is processed by the DLMS, it is loaded into DSP1 because of the same algorithm, as shown in Fig. 4D.

Thereafter, load L4 from DSP4 and load L2 from DSP2 are processed in the DSP4 and DSP2, respectively, and they are unloaded from their respective DSPs upon completion of processing at the end of the frame, as shown in Figs. 4E-4F.

From the previous example, the DLMS distributes additional loads to one or more DSPs so that each DSP is processing efficiently and in unison. The DLMS may split loads between two or more DSPs or may load the additional load into one DSP, so long as the entire system is efficiently utilized. The DLMS allows the overall SOC device to function as efficiently as possible by distributing and/or re-distributing loads between multiple DSPs.

The DLMS can also predict the type of load and algorithm of arriving data so that they can be distributed appropriately to the multiple DSPs. The DLMS also allocates and distributes the new load uniformly among the several DSPs in order to reduce the time skew. In other words, the DLMS includes information relating to the internal characteristics of the DSPs. For example, the DLMS has information regarding the number of DSPs, their capacities (running at 200 MHz, processing capacity in 20 MIPS), present loading status (algorithms 1 is used in DSP 1, 2 and 3, and algorithm 2 is used in DSP 4, thus no resetting required), algorithms deployed and allowed, and algorithm to DSP preference or relationship table (know which algorithm to load to a particular DSP by looking up on a table). Figs. 5A-5C illustrate an example of the DSP MIPS budget using three DSPs and two algorithms with data being fed at predetermined time intervals of, for example, 10 milliseconds. When an SOC device has n DSPs, and the maximum load is LM, the average load per DSP for uniform loading is assumed to be LM/n. Also, the minimum additional DSP MIPS budget to allow for two algorithms switching is

LM/n and the additional loading per DSP for uniform loading is now LM/n(n-1).

Thus, using the assumptions stated above, for n=3, the uniform loading for DSP MIPS budget is LM/3 + LM/(3)(2), which is equivalent to 0.5LM. The previous example illustrates that at least 50% LM of DSP loading is needed per DSP. Further, it is assumed that MIPS reserve for non-uniform loading and overhead margin is generally 20%, which is equivalent to 0.6LM per DSP instead of 0.5LM, as calculated above.

Figs. 4G-4H illustrate the intra-transaction DSP algorithm switch optimization in accordance with the preferred embodiment of the present invention. Using a specific example, Fig. 4G illustrates a table having three DSPs and 4 running algorithms, and Fig. 4H illustrates a simplified time diagram of the DSPs processing times of the DSPs in Fig. 4G. In this example, there is an assumption that there are 24 calls of which 15 calls are running algorithm 1, 5 calls running algorithm 2, 2 calls running algorithm 3, and 2 calls running in algorithm 4. The 15 calls will be processed by the DSP1 running algorithm 1. DSP1 will have the longest processing time since it needs to process the most calls. The 5 calls will be processed by the DSP2 running algorithm 2, which processing time is less than DSP1. The 2 calls will be processed by DSP3 running algorithm 3, which processing time is less than DSP1 and DSP2. Since DSP3 processing time is the shortest, the final 2 calls are processed by DSP3 after switching such that algorithm 4 is used to process these final 2 calls. This optimization scheme allows all three DSPs to process the calls simultaneously while intelligent switching decisions are performed by the DLMS such that all three DSPs are working efficiently on one chip.

Fig. 5A illustrates the maximum load per DSP as being 0.6LM for a total DSP MIPS of 1.8LM for the case of three DSPs and two algorithms (switching once). When

uniformly loaded, each DSP can be loaded similarly (e.g., 0.3 LM). In Fig. 5B, when switching is required in uniform loading, the load in DSP1 is re-distributed evenly and loaded into DSP2 and DSP3, thereby providing loads of 0.45LM to DSP2 and DSP3. The new load is then loaded into the DSP1. Fig. 5C illustrates a condition when there are non-uniform loading and arbitration delays associated with the system. In this case, DSP2 and DSP3 do not have the same loads therein.

Figs. 6A-6C illustrate an example of the DSP MIPS budget using three DSPs and three algorithms with data being feed at intervals of, for example, 10 milliseconds. Using the same formulation discussed above, the maximum load per DSP is 1LM, which equates to a maximum load of 3LM for the three DSPs and three algorithms (switching twice). This is illustrated in Fig. 6A. Fig. 6Bi illustrates tables after the second algorithm is loaded into the DSP1. In this case, the load that was originally in the DSP1 is flushed or cleared out and is loaded into the DSP2 and DSP3 in equal loads. This allows the new load with the second algorithm to be loaded into DSP1, as shown in Fig. 6Bi. Thereafter, when a third algorithm is inputted into the DSP array, the load in the DSP2 is again flushed or cleared out and is loaded into DSP3. The load having the third algorithm is then loaded into DSP2, as illustrated in Fig. 6Bii. In this example, DSP1 includes loads corresponding to the second algorithm, DSP2 includes loads corresponding to the third algorithm, and DSP3 includes loads corresponding to the first algorithm. Additionally, Fig. 6C illustrates tables that corresponding to conditions with non-uniform loading and arbitration delays.

Assume that there are more algorithms than there are DSPs, then optimization is required intra-transaction between the DSPs. Using the example where there are 5

algorithms and 4 DSPs for simplicity. Assume that there are 24 T1 channels and all DSPs can handle the same amount of MIPS. DSP1 takes 15 calls, DSP2 takes 5 calls for algorithm 2. The smaller loads of 2 for algorithms 3 and 4 are sent to DSP3. Thus, the DSP3 switches once. Time wise during transaction time DSP1 takes time a, DSP2 takes time b and DSP 3 takes time c including the switching time for the two algorithms. This invention allows switching within DSPs when one DSP has to handle more than one algorithm.

From the previous examples, one skilled in the art can implement any number of DSPs and different algorithms to efficiently distribute and redistribute loads between the DSPs using the DLMS. The DLMS can process both voice channels (i.e., 30 channels) and modem data channels (i.e., 16 channels) simultaneously in a single or multi-user mode. The DLMS balances the loads in order to effectively and efficiently utilize all the available resources in the system. Preferably, dynamic load balancing decisions are based on current assignment at load arrival times to achieve an equal balance of the load although there may be high load variations or load surges.

The DLMS can also achieve both inter- and intra-transaction parallelism or concurrent execution of independent transactions. For example, the DSP1 can process one part of the transaction and DSP2 can process another part of the transaction. . Each DSP associated with the DLMS can have 8 stage pipeline parallelism. The DLMS also assumes that the input data queue will be stored in a DRAM/Data cache, which data is staged at intervals less than 80 TDM frames to reduce transaction response time. When the DLMS works in conjunction with the MT MMS, they can support high degree of inter/intra transaction parallelism from shared on/off chip high-speed external memory.

As discussed above, one of the important features of the DLMS of the present invention is its ability to switch algorithms from one DSP to another. The DLMS includes functionality that allows for predictive or forward-looking action for early protocol switch-warning, data queue optimization and latency trade off. In case of a very high throughput system having a large DSP array, the DLMS must be capable of interfacing directly with the memory system or multiple CPUs to control the data delivery to the DSP array. Preferably, the DSPs used in the present invention should be capable of processing a 24-channel T1 span of VoIP (Voice over IP) or FoP (Fax over IP) per processor or up to 16-channels of modem data.

Each DSP can function as a coprocessor to a CPU. This enables the CPU to transmit data/instructions and "start conditions" to the DSP in a more efficient manner. The CPU then receives the data/instructions when the DSP is done. In this manner, data/instructions are sent back and forth between the CPU and the DSP when the CPU executes the appropriate instructions. The DSP can also issue an interrupt signal to the CPU and alter the execution flow of the CPU.

There are generally two commands (instructions) for loading/storing data between main memory and the general registers of the coprocessors. The memory can be either on the CPU bus or internally in RAM on the cache bus. Because the coprocessor cannot stall the CPU, read and write signals have to be completed within one clock cycle. The CPU can also load and store to these registers either from memory or from ALU registers. Preferably, the coprocessor instructions perform single word transfer to/from CPU register to DSP register.

Fig. 7A illustrates a schematic diagram of a co-processor interface (CI) of the

DLMS, CPU, and DSP in accordance with the preferred embodiment of the present invention. The signal definitions are now described for a more complete understanding of the present invention: Z represents the DSP number; CZWR_addr represents the write address for coprocessor interface Z; CZWR_gen represents the select signal for general register of processor Z; we0 represents the write enable of register 0; we31 represents the write enable of register 31; CZRD_addr represents the read address for processor Z; CZRD_con represents the read coprocessor Z or general register; CZWR_data represents the write data to coprocessor Z; and CZRhold represents coprocessor Z must hold the previous value of read data on the read data bus.

The coprocessor interface (CI) 70 can support up to 32 general registers 76, 32 control registers 86, and a control flag. In greater detail, a CZWR-addr signal is transmitted from the CI 60 to a write address decoder 72, which in turn generates signals for inputting into AND gates 74a, 74b,..., 74n. In addition, CZWR-gen signals are also inputted into the AND gates 74a, 74b,...74n. The outputs of the AND gates we0, we1,...we31, are then inputted in the general registers 76 having 32 general registers along with the CZWR-data signal from the CI 70.

During operation, the CPU via the coprocessor interface issues a write address CZWR_addr. The address is decoded in the write address decoder 72. A select signal for general register for the processor CZWR_gen along with the write address CZWR_addr are inputted in the AND gates 74a, 74b, 74c for generating write enable signals we0, we1, we31. The write enable signals we0, we1, we31, or communication data is then written in the registers using the write data to coprocessor signal CZWR_data. Data from the general registers is then transmitted into a first MUX 78 and outputted to a second MUX

84. An output from the control registers 86 works in the same manner as described above. The read address decoder 80 reads the write address CZWR_addr along with the CZRhold signal and then issues a read address to the MUX 78 via the delay 82. Further coupled to the MUX 84 are control registers 86 having 32 control registers.

Fig. 7B illustrates an example of an implementation of the general and control registers 76, 86 using multiple MUXs and flip-flops.

Figs. 8A-8B illustrate multi-coprocessor interfaces that can be implemented in accordance with the preferred embodiment of the present invention. In Fig. 8A, the coprocessor interface (CI) 112 is interfaced between a CPU 110 and three DSPs 114a, 114b, 114c. The CI 112 includes three ports C1, C2, C3, where each port is used for communicating with each DSP 114a, 114b, 114c, respectively. Preferably, the communication link between the CI 112 and each DSPs 114a, 114b, 114c, can accommodate 64 I/O data, 5 Raddr, 5Waddr and other controls. Fig. 8B illustrates an CI 122 interfaced to the CPU 110 similar to the manner described in Fig. 8A. The CI 122, however, includes ports C1, C2, C3, where each port communicates with at least two DSPs. For example, port C1 communicates with DSP array 124a, port C2 communicates with DSP array 124b, and port C3 communicates with DSP array 124c.

Fig. 9A illustrates a system including a CPU, CI, DLMS, MT-MMS, memory access system, and multiple DSPs in accordance with the preferred embodiment of the present invention. As illustrated, the CI 132 is interfaced between the CPU 110 and the DLMS 134. The DLMS 134 communicates with DSP array 144a, 114b, 144c via ports similar to the ports C1, C2, C3 in the CI 132. Also included in this system is an MT MMS 136, which communicates directly with the CPU 110 and the DLMS 134 for

managing memory requests. In other embodiments, the CPU 110 can be connected to memory access controllers (MAC) via M bus channel and a P bus controller as illustrated in Fig. 12. This effectively converts the shared memory between the multiple DSP systems to effectively shared nothing (SN) DSP to six separate memory sections threaded by separate memory access controller threads.

The DSPs 1 - 6 can be programmed as a sea of gate machines (i.e., they must not all be of the same kind). The DLMS 134 will give each DSP tasks that best matches its capabilities. For example, one DSP may be a 16 x 16 array of multiply and accumulate MAC devices. The DLMS 134 will then extract the algorithms or the parts of the algorithms that are suitable for massive parallel processing and assign them to this particular DSP having the 16x16 MAC array. It will also thread the results into it from another programmable DSP and thread the results out of it into another programmable DSP. The point here is that there are several levels of optimization possible in implementing several algorithms in parallel such as (1) intra-transaction optimization, (2) inter-transaction optimization, and (3) inter-algorithm/sample processing optimization. This includes selecting parts of the sample algorithm to thread out to parallel sea of gate array DSP. The interaction between the CPU and the DLMS hardware is designed to achieve dynamic (i.e., in real time) optimization of the load assignment between the DSPs. The sea of gate DSP with an array of MACs may have much more restrictive number of DSP instructions than a programmable DSP.

Fig. 9B illustrates an effective behavior model of the system in Fig. 9A in accordance with the preferred embodiment of the present invention. Effectively, the system illustrated in Fig. 9A can be shown as a CPU communicating with multiple DSPs

on a very wide bus, with the DSPs communicating with multiple external memories on another very wide bus.

Fig. 10 illustrates an MT MMS loading DSP with high frequency data in accordance with the preferred embodiment of the present invention. The wide memory is 4x16 or 4x32 bits. The registers in the DLMS 100 can be loaded from the CPU or from external memory. The CPU may provide the data from the address latch and can read from the wide bus. The DLMS registers are written from data via the MUX 190a, 190b. The signals sent to the MUX 190a, 190b, include data from the wide memory bus, CZWR_data, Cbit, and Address latch. The DSPs 12a, 12b, 12n communicates with the DLMS 100 to access data in the registers.

Fig. 11 illustrates an example of a two threaded memory which is used with two memory access controllers (MACs) in accordance with the preferred embodiment of the present invention. In lieu of using the MT MMS as described earlier herein, the present invention can be implemented with multiple MACs. Each channel is connected to a DMA and memory bus (M Bus). In the first MAC1 1100, DSP 1 is connected to channel 1, DSP 2 is connected to channel 2, DSP 3 is connected to channel 3, and DSP 4 is connected to channel 4. In the second MAC2 1110, DSPs 1-4 are connected to channels 1-4, Ethernet transmit (E Net TX) is connected to channel 5, Ethernet receive (E Net RX) is connected to channel 6, and communication cores are connected to channel 7. In this manner, DSP1 and DSP2 can access the MAC1 1100 via channels 1 and 2, while DSP 3 and CPU can access the MAC2 1110 via channels 1-4 for simultaneous data processing.

To better describe this figure, assume that DSP 1 requires the bandwidth of 40 to 60 Mb/sec and the CPU requires 80 to 100 Mb/sec and the bus between the MACs and

the memory provides only 100 Mb/sec (after averaging). Then, only two MACs as illustrated herein can handle both processors at the same time. Compared to a system having the MT-MMS with arbitration capabilities, the CPU in Fig. 11 has to make sure that data is not routed to the same MAC.

Fig. 12 illustrates a system including a CPU connected to memory access controllers in accordance with the preferred embodiment of the present invention. There is a separate peripheral control bus controller 220 and memory bus channel 210 between the CPU and the MACs 230a, 230b. Depending on the address being issued, the control bus controller 220 can select the proper MAC via signals SEL1 and SEL2.

Next, Fig. 13 illustrates a specific implementation of DMA and MAC clusters in accordance with the preferred embodiment of the present invention. Again, it is important to note that other system architectures and components can be used in the present invention than those described herein. The request signal for a particular MAC is sent via signals Mb_blk_Request and address range qualifier. Based on the address sent, the MAC1 or MAC 2 is selected for reading/writing data. What is important to note from this diagram is that there are separate data buses for receiving data in a simultaneous manner.

Fig. 14 illustrates a diagram of multiplexing the data bus to individual channels in accordance with the preferred embodiment of the present invention. The data from the MACs is preferably a 32 bits wide and is sent to the multiple AND gates. Each bit is coming from either MAC1 or MAC2. Depending on which channel is active (qual), one of the AND gates give an output and into the right channel. For each of the CPU, DSP, DLMS, and communication cores, there is data coming from two data bits to generate the

final data output into the channel X.

Fig. 15 illustrates a flow chart implementing the real time load management system in accordance with the preferred embodiment of the present invention. As shown and discussed earlier herein, the present invention can be used for optimizing (1) selection of the transaction/frame time, (2) inter-transactions, (3) intra-transactions, (4) sub-frame time selection, and (5) hardware mapping of a vocoder algorithm to process a sample.

Fig. 16 illustrates a block diagram of the DSP load management system parallel word transfer in accordance with the preferred embodiment of the present invention. The DLMS includes a decoder capable of decoding CPU instructions that require parallel data word transfers between DSPs and between DSPs and memories. Data from DSP1 can be sent to a first decoder and after decoding sent to the DSP31. Likewise, data from DSP2 can be sent to a second decoder and after decoding sent to the DSP1. Further, data from DSP3 can be sent to a third decoder and after decoding sent to the DSP2. The process is based on decoding of CPU instruction containing an opcode or a bit set by the CPU in the control registers of the DLMS. This process can be performed simultaneously with multiple DSPs.

Fig. 17 illustrates a block diagram of the PE array controller in accordance with the preferred embodiment of the present invention. Referring back to Fig. 9A, the DLMS/PE arrays can be substituted for one or more DSP array 144a, 144b, 144c. In this manner, the DLMS/PE array is a smaller array having less captive memory. Subroutines are fed in parallel to the PE/+ addition, PE/X multiply, and PE/- subtraction through internal registers in the DLMS. After performing the subroutines using the PE/+, PE/X,

and PE/-, the data is sent back to the DLMS/PE array.

Figs. 18A-18C illustrate a system having a configurable bridge/gateway and/or switching functionality for optical networks in accordance with the preferred embodiment of the present invention. Fig. 18A illustrates a configurable bridge/gateway for an optical network using the DLMS and MT MMS. The MT MMS 320 communicates with CPU subsystems 350, DLMS 360 and DSP subsystems 370a, 370b, 370c. In addition, the MT MMS communicates with on/off chip memories 340a, 340b, 340c, 340d, and the DLMS 360 is interfaced between the CPU subsystems 350 and DSP subsystems 370a, 370b, 370c. When data is inputted into the system, the S/P converter 310 converts the data, whose rate is preferably between 0.5-2.5 Gbps. After the converted data enters the MT MMS, it is processed appropriately, as discussed above, and then sent out via the P/S converters 330 at a rate that is preferably between 0.5-2.5 Gbps. Fig. 18B illustrates the switching rates compatible with the present invention. Data can be inputted at a rate of 16x622 Mbps or 64x160 Mbps to the switch 600. Thereafter, the data is outputted using the switch 600 at a rate of 12x1.25 Gbps or 4.x2.5 Gbps.

Fig. 18C illustrates a more detailed diagram of the configurable bridge/gateway in accordance with the preferred embodiment of the present invention as shown in Fig. 18A containing a multi-threaded memory access controller, MT-MMS, DLMS, CPU and DSPs. Router 1 and router 3 can input data into the bridge/gateway before such data is inputted in router 2 and router 4. This short reach (i.e., same "campus") bridge is used to interface between routers using low cost fiber and laser system. The DSP systems can be used for forward error correction and IP to Sonet encoding. As illustrated, the bridge/gateway includes functionalities such as optical IP MUX/DEMUX, IP to Sonet

Encoder/Decoder, Optical gateway with statistical multiplexing, and fiberoptic link transceiver. These functionalities allow data from ATM, IP, T1/E1 to be transmitted to the router 2 and router 4 at higher data rates. This allows conversion from IP to sonet. Routers break down the data into smaller frequencies and the switch allows the routers to communicate with each other.

In the previous descriptions, numerous specific details are set forth, such as specific functions, components, etc., to provide a thorough understanding of the present invention. However, as one having ordinary skill in the art would recognize, the present invention can be practiced without resorting to the details specifically set forth.

Although only the above embodiments have been described in detail above, those skilled in the art will readily appreciate that many modifications of the exemplary embodiments are possible without materially departing from the novel teachings and advantages of this invention.

I claim:

1. A system for providing parallel processing of data to a plurality of digital signal processors (DSPs), comprising:

means for transmitting communication data to a load management system from a CPU;

means for selecting a digital signal processor (DSP) from a plurality of DSPs for processing the communication data;

means for processing the communication data using the selected DSP; and

means for transmitting the processed data back to the CPU and to a communication device.

2. A system of claim 1, wherein the communication data is transmitted from a VoIP medium.

3. A system of claim 1, wherein the communication data is transmitted from a FoP medium.

4. A system of claim 1, wherein the communication data is transmitted from an IP to sonet medium.

5. A system of claim 1, wherein the communication data is transmitted from an encoder/decoder.

6. A system of claim 1, wherein the communication data is transmitted from a broadband communication medium.

7. A system of claim 1, wherein the communication data is transmitted from an image processing medium.

8. A system of claim 1, wherein the communication data is transmitted from

a data modem.

9. A system of claim 1, wherein the load management system comprises:

a plurality of direct memory access (DMA) devices having internal registers, a plurality of FIFOs, a plurality of state machines associated with the plurality of FIFOs, and a memory interface for interfacing the plurality of DMA devices with an external memory device;

a plurality of status and controls registers coupled to the plurality of DMA devices;

a CPU interface for interfacing the CPU with the plurality of status and control registers; and

a DSP interface for interfacing the plurality of DSPs with the plurality of DMA devices.

10. A system of claim 9, wherein the DSP interface includes a program/data memory and a ping-pong memory.

11. A system of claim 9 further comprising an external memory, wherein the external memory is coupled to the plurality of DSPs through dedicated memory threads.

12. A system of claim 9, wherein the CPU interface includes a routing MUX, wherein the routing MUX is coupled to the external memory device.

13. A system of claim 12, wherein the external memory device comprises a memory access controller array.

14. A system of claim 12, wherein the external memory device comprises a memory management system.

15. A DSP load management system, comprising:

a plurality of direct memory access (DMA) devices having internal registers, a plurality of FIFOs, a plurality of state machines associated with the plurality of FIFOs, and a memory interface for interfacing the plurality of DMA devices with an external memory device;

a plurality of status and controls registers coupled to the plurality of DMA devices;

a CPU interface for interfacing a CPU with the plurality of status and control registers; and

a DSP interface for interfacing a plurality of DSPs with the plurality of DMA devices.

16. A DSP load management system of claim 15, wherein the external memory device comprises a memory access controller array.

17. A DSP load management system of claim 15, wherein the external memory device comprises a memory management system.

18. A DSP load management system of claim 15 further including a decoder for decoding instructions from the CPU.

19. A DSP load management system of claim 18, wherein the instructions include load and algorithm switching instructions in the plurality of DSPs.

20. A DSP load management system of claim 19, wherein the load and algorithm switching instructions provides optimal processing of the plurality of DSPs.

21. A method of providing parallel processing of data to a plurality of digital signal processors (DSP), the method comprising:

transmitting communication data to a central processing unit (CPU);

writing the communication data to a selected register from a plurality of registers in a load management system, wherein each register is coupled to a DSP via a designated thread;

transmitting the communication data from the selected register to its corresponding DSP via the designated thread; and

processing the communication data using the corresponding DSP.

22. A method of claim 21, wherein the register is selected based on the current load status of the plurality of registers.

23. A method of claim 22, wherein the register is selected based on the algorithms loaded in the plurality of registers.

24. A method of claim 21, wherein the communication data is loaded into the register having the least load and same algorithm as the new load.

25. A method of claim 21, wherein the communication data is loaded into the register after switching the load from the register into a second register.

26. A method of claim 21, wherein the plurality of registers includes at least two different algorithms.

27. A method of claim 21, wherein the plurality of registers includes at least three different algorithms.

28. A method of claim 21 further comprising simultaneously processing data using the plurality of DSPs.

29. A method of claim 21, wherein the communication data is transmitted from one of a VoIP medium, a FoP medium, an IP to sonet medium, an encoder/decoder, a broadband communication medium, an image processing medium, and a data modem.

30. A method of selecting a particular DSP from a plurality of DSPs for loading a new load, the method comprising:

analyzing the current loads and algorithms associated with each DSP; and

selecting the particular DSP from the plurality of DSPs based on the current loads and algorithms, wherein the selected DSP has a least load and includes the same algorithm as the new load.

31. A method of optimizing the processing capabilities of a plurality of digital signal processors (DSPs) on an SOC device having a DSP load management system (DLMS), comprising:

transmitting communication data to a central processing unit (CPU);

writing the communication data to a selected register from a plurality of registers in a load management system, wherein each register is coupled to a DSP via a designated thread;

transmitting the communication data from the selected register to its corresponding DSP via the designated thread, wherein the corresponding DSP is the least loaded DSP on frame bourndaries; and

processing the communication data using the corresponding DSP.

32. A method according to claim 31, wherein the DLMS provides inter-transaction optimization, intra-transaction optimization, and hardware mapping optimization.

ABSTRACT

A highly intelligent DSP load management system is described herein for enhancing the processing capabilities of an SOC device. The DSP load management system enables parallel processing of data at high frequency and distributes, reads and writes data to several CPUs and/or DSPs in the same clock cycle. In addition, the DSP load management system provides forward looking real-time evaluation of arriving data and diverts tasks from one DSP to another, with short or zero latency. The DSP load management system is interfaced between one or more CPUs, one or more DSPs and/or a memory management system for enabling parallel processing of data at high frequency.

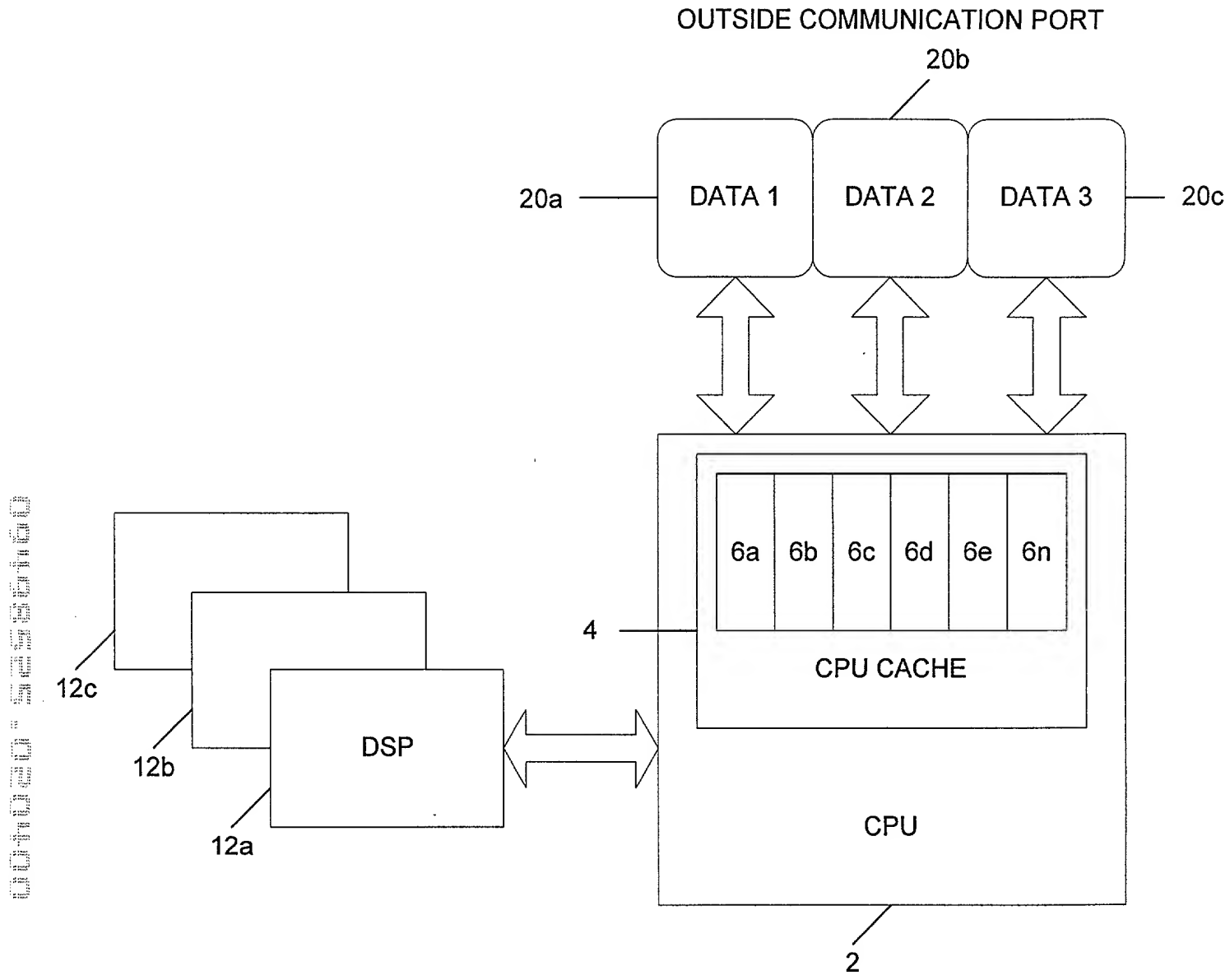


FIG. 1 (PRIOR ART)

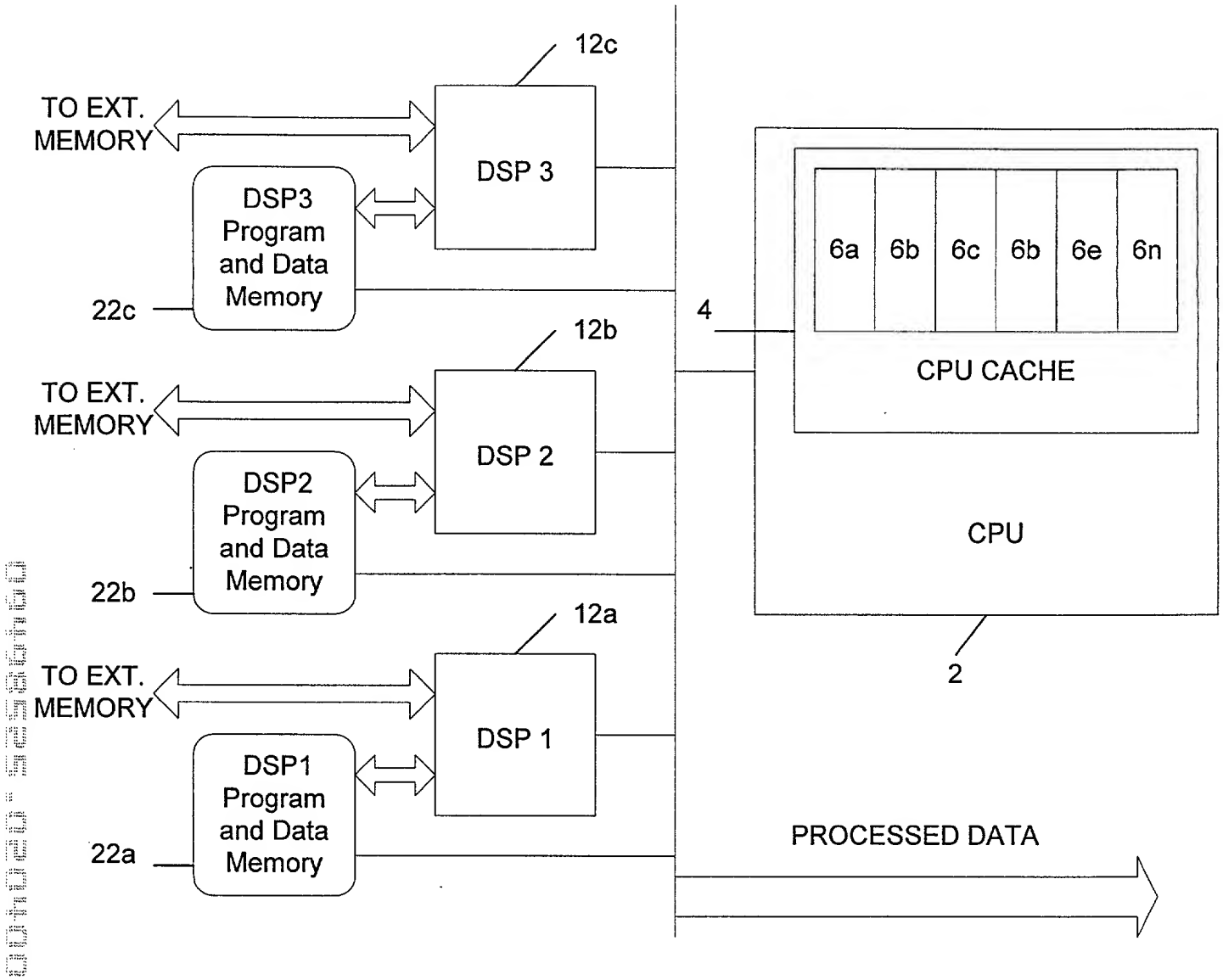


FIG. 2 (PRIOR ART)

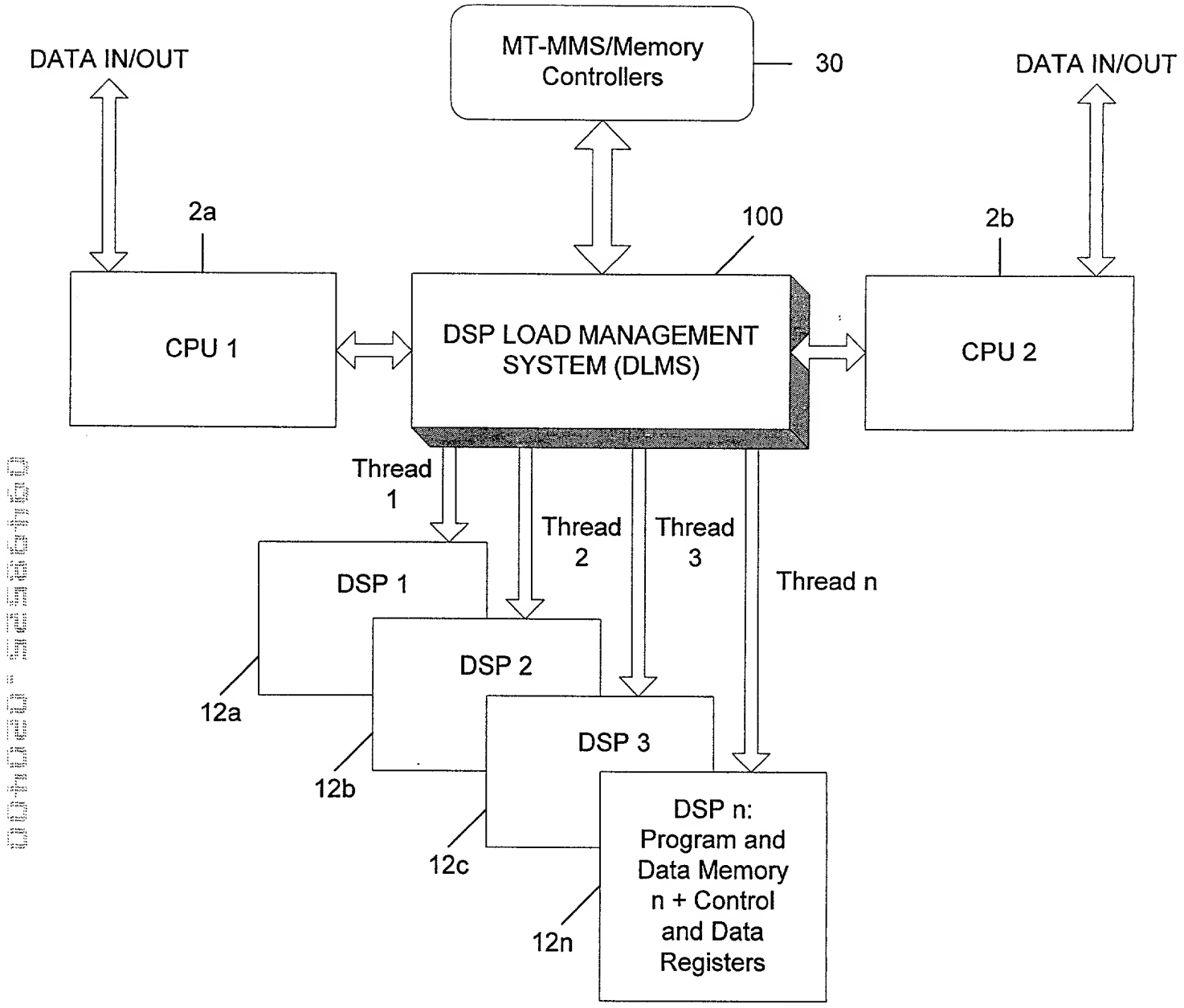

DATA IN/OUT
MT-MMS/Memory Controllers
30
DATA IN/OUT
2a
100
2b
CPU 1
DSP LOAD MANAGEMENT SYSTEM (DLMS)
CPU 2
Thread 1
Thread 2
Thread 3
Thread n
DSP 1
DSP 2
DSP 3
12a
12b
12c
12n
DSP n: Program and Data Memory n + Control and Data Registers

FIG. 3A

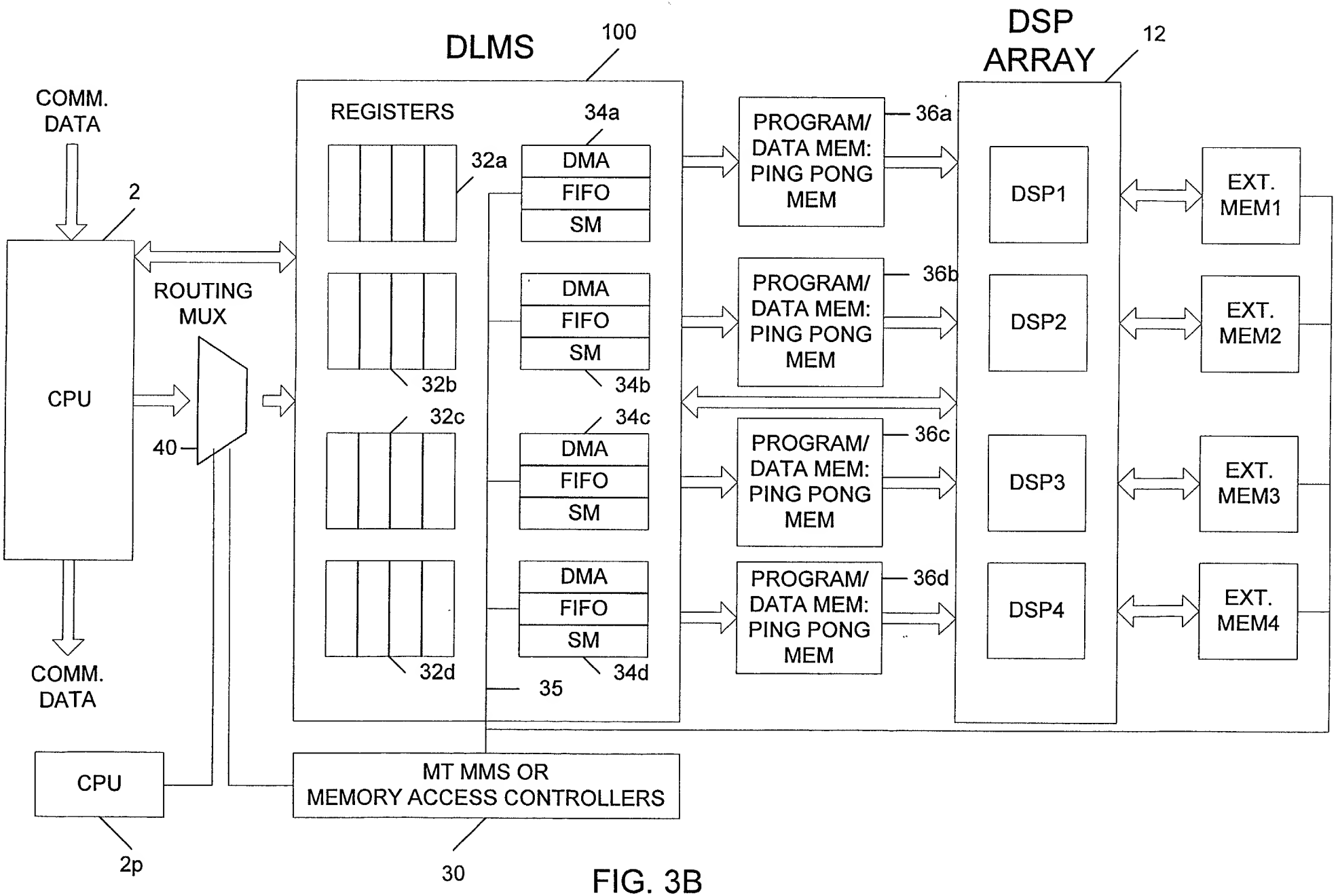
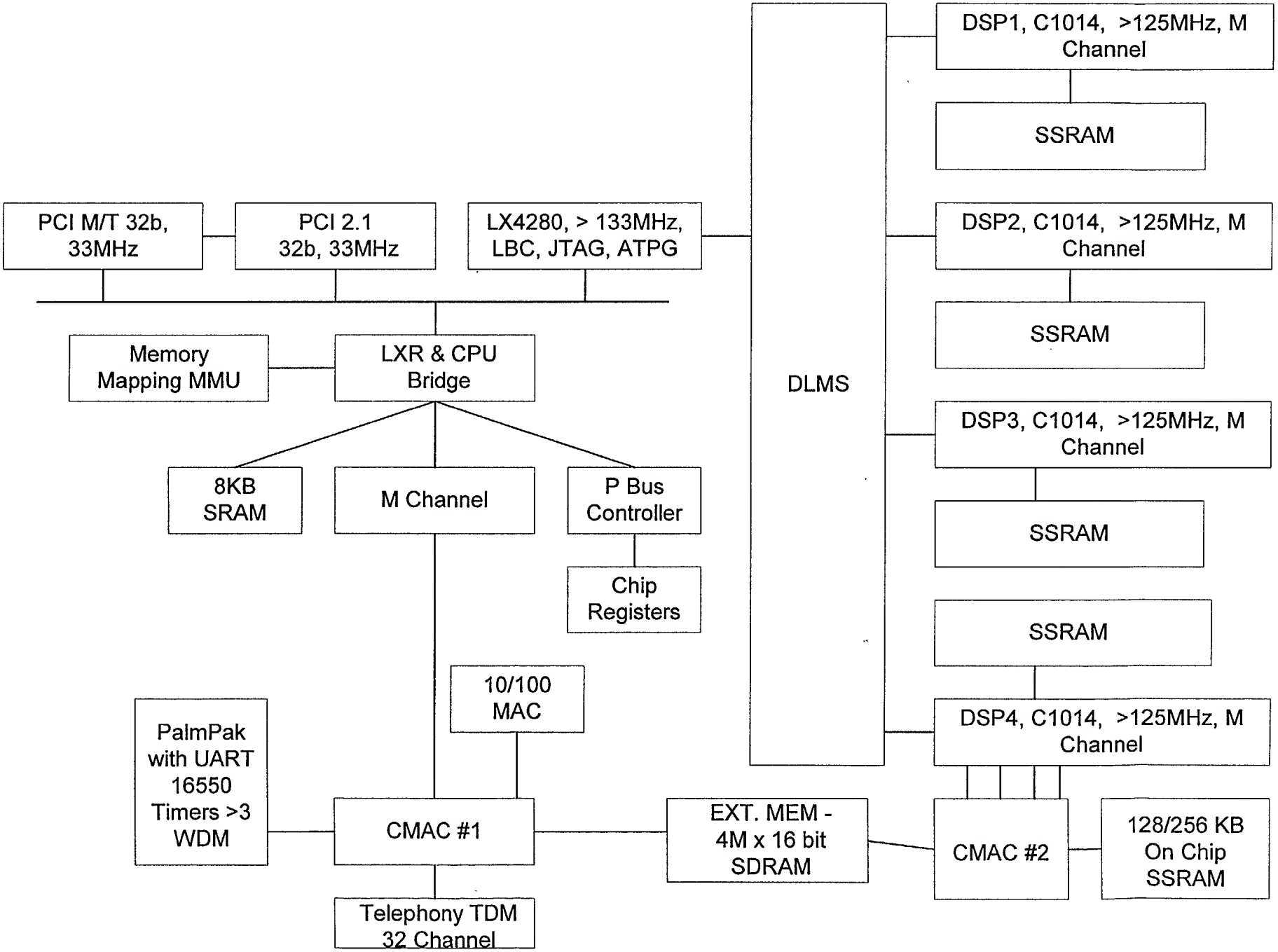


FIG. 3B

FIG. 3C



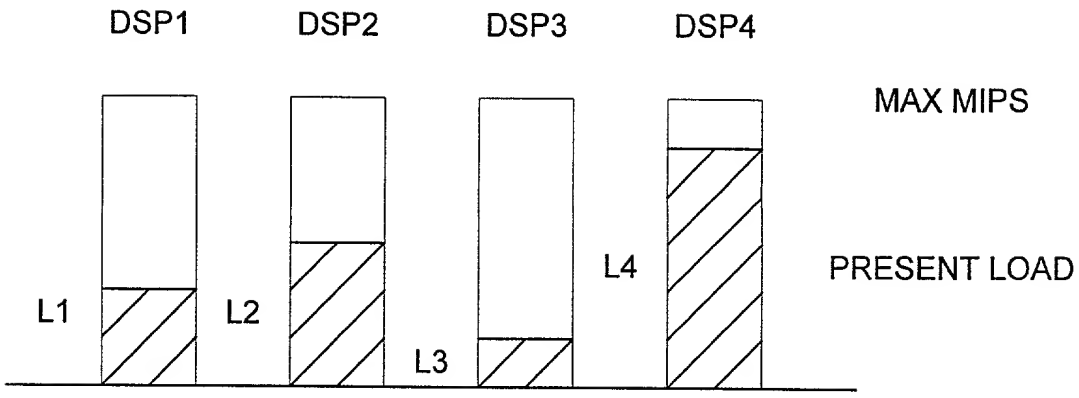


FIG. 4A

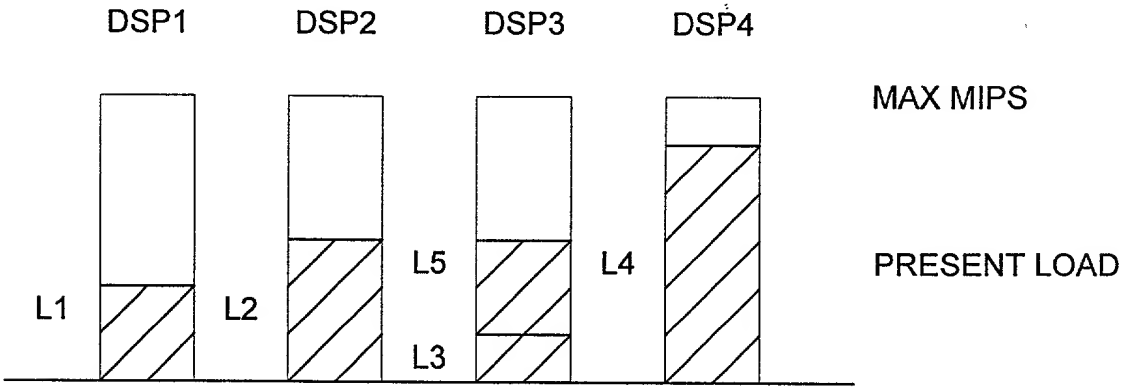


FIG. 4B

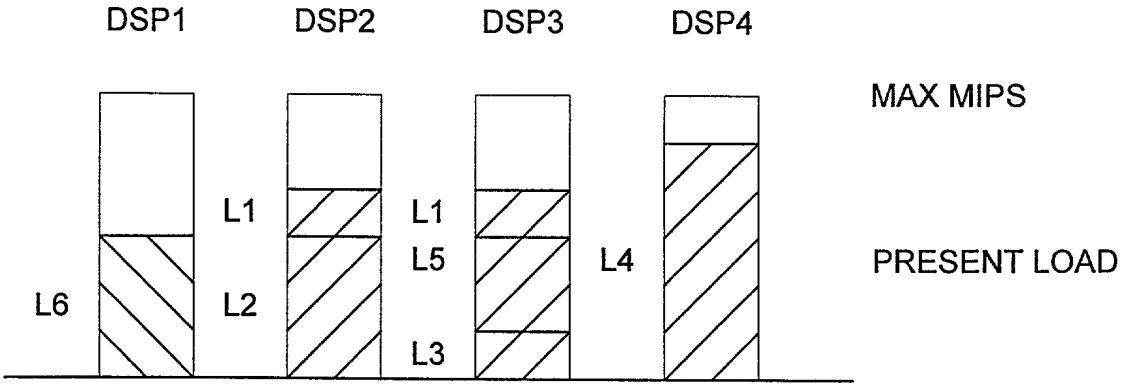


FIG. 4C

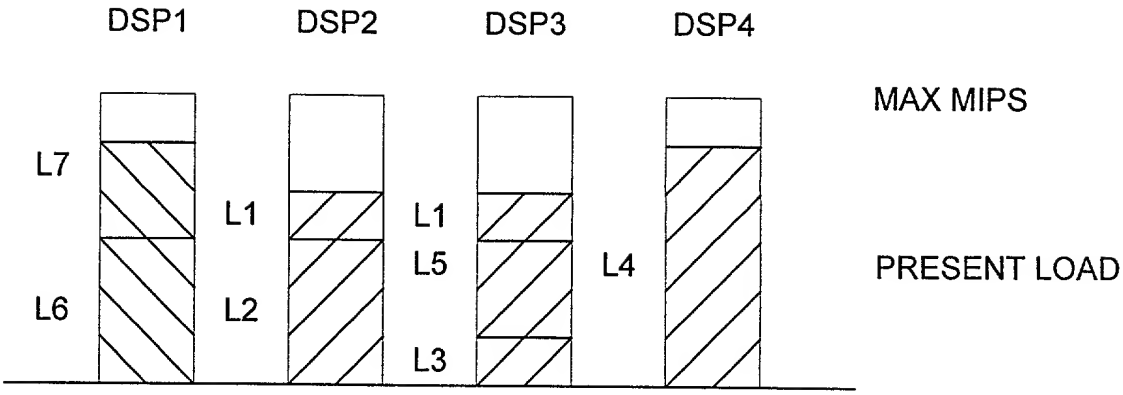


FIG. 4D

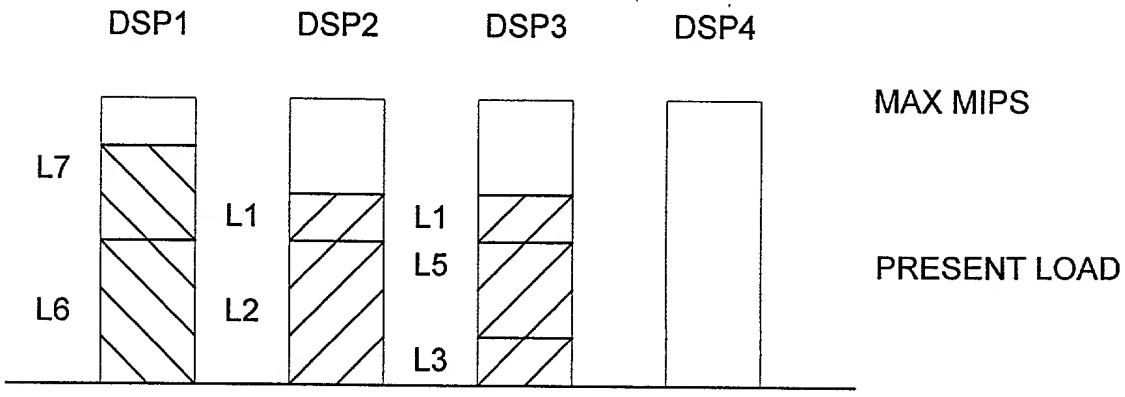


FIG. 4E

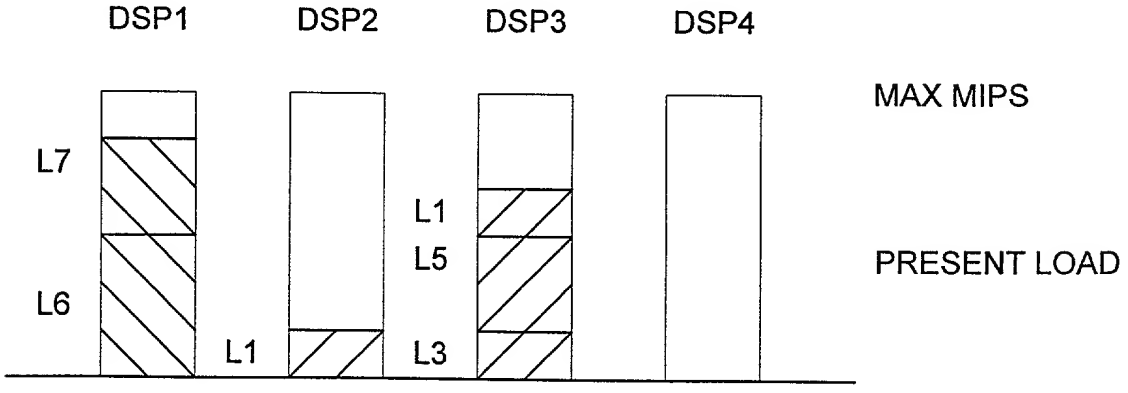


FIG. 4F

	DSP1	DSP2	DSP3	
ALGORITHM 1	15			
ALGORITHM 2		5		
ALGORITHM 3			2	
ALGORITHM 4			2	

FIG. 4G

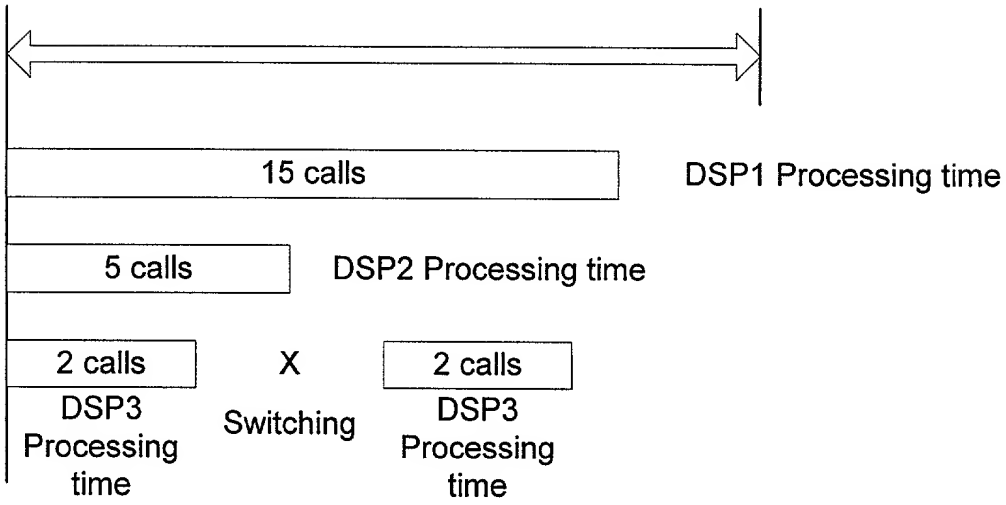


FIG. 4H

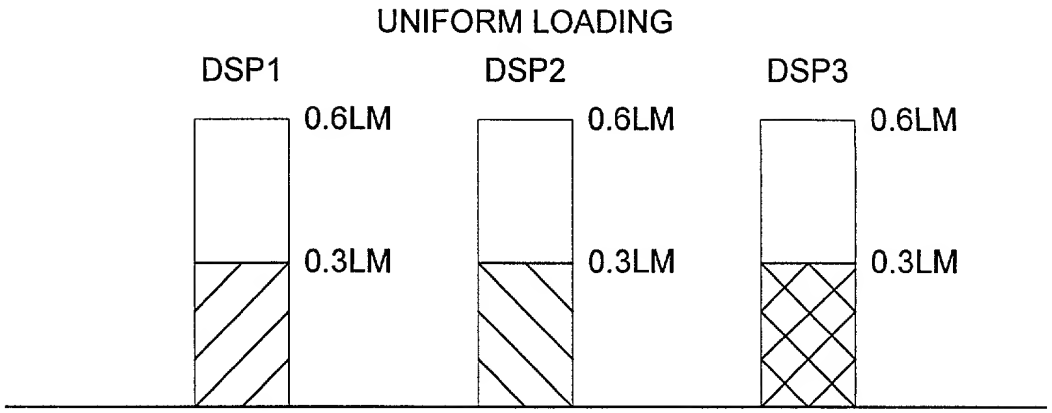


FIG. 5A

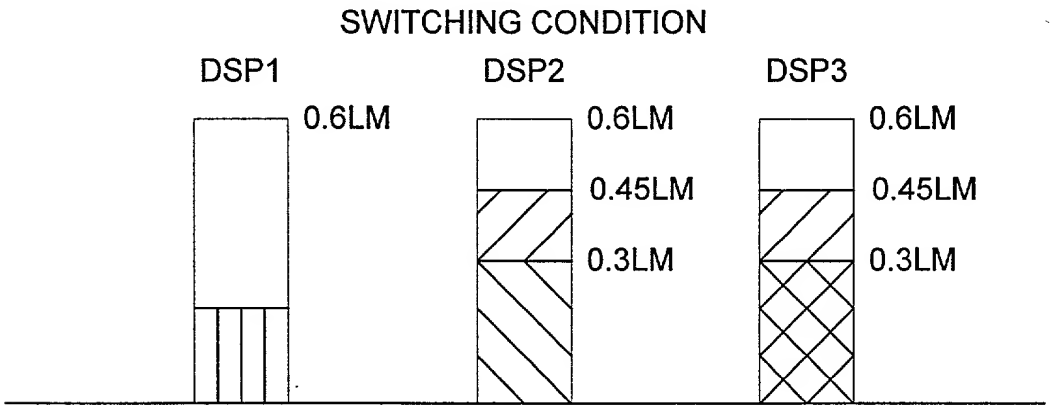


FIG. 5B

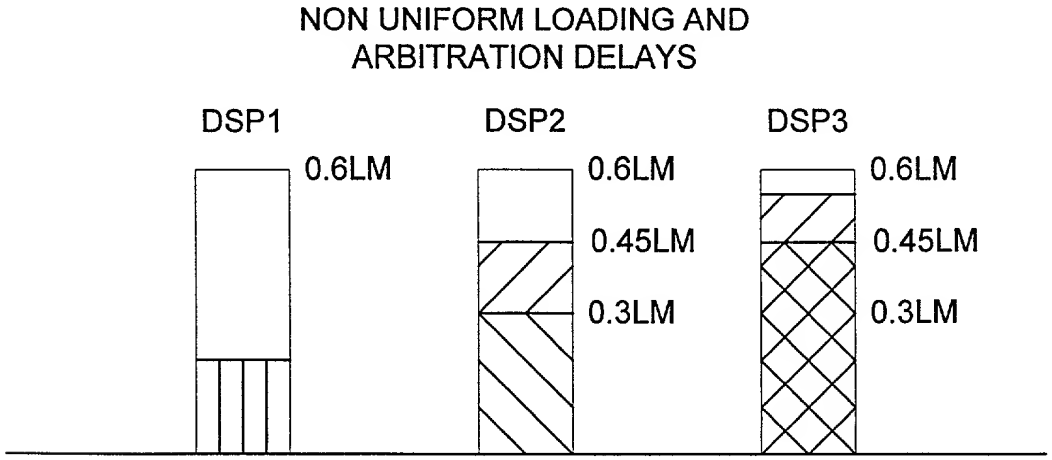


FIG. 5C

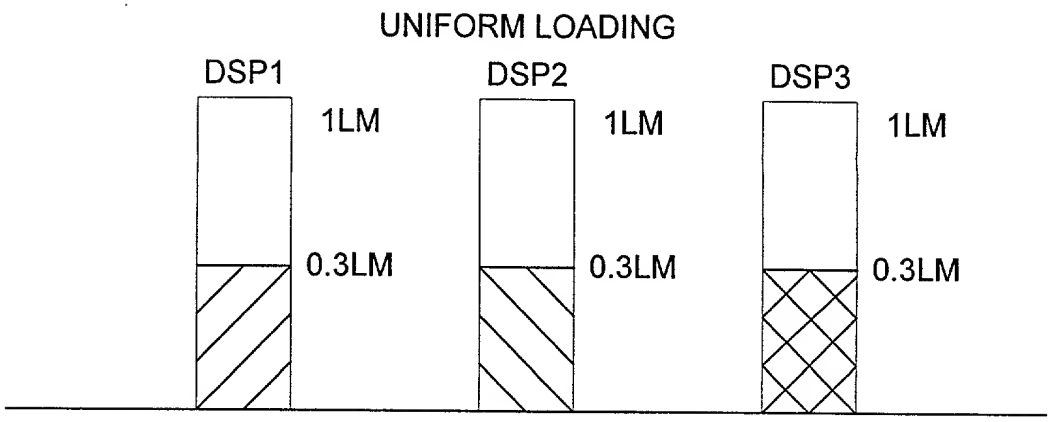


FIG. 6A

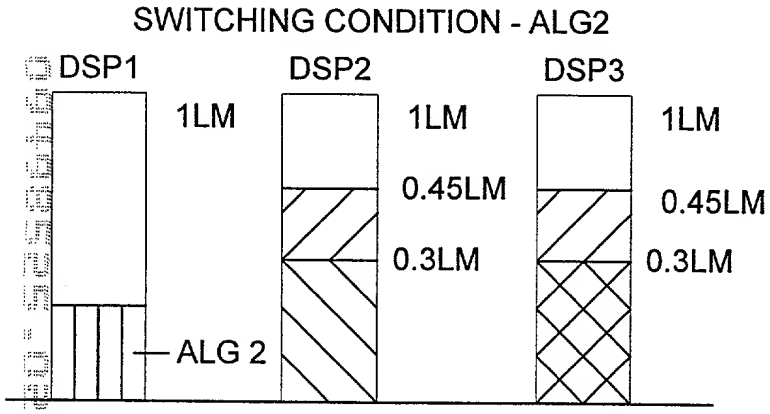


FIG. 6Bi

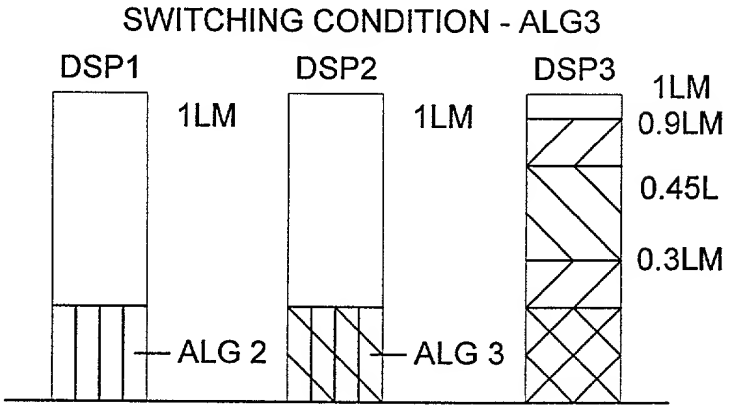


FIG. 6Bii

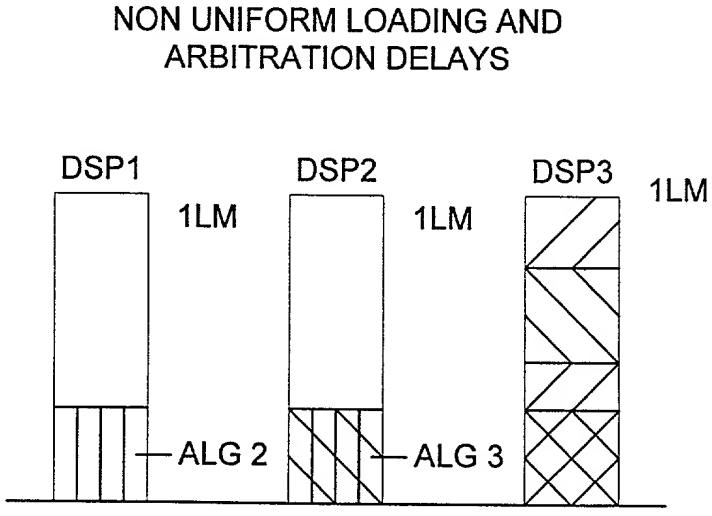
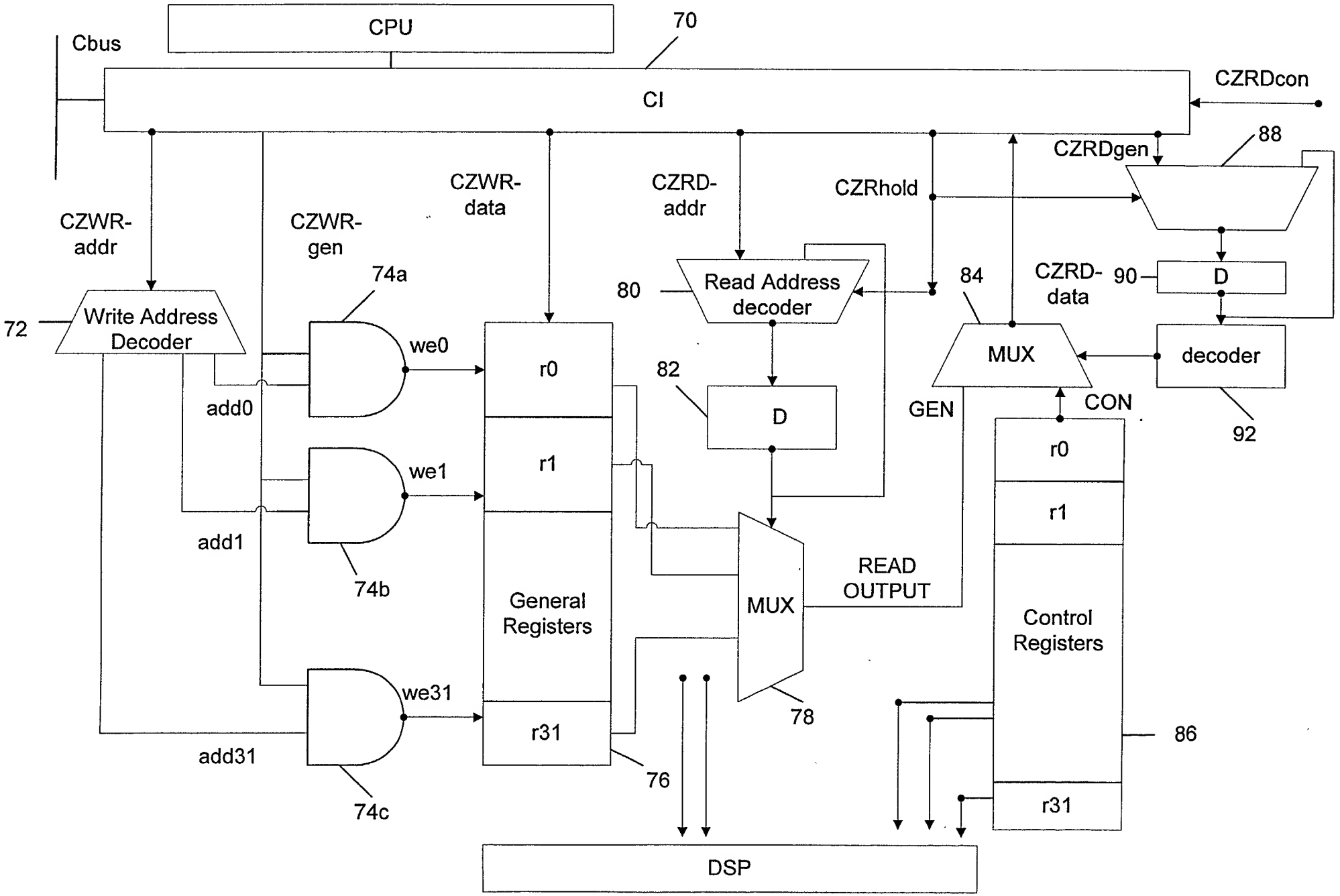


FIG. 6C

FIG. 7A


CPU
70
Cbus
CI
CZRDcon
CZRDgen
88
CZWR-addr
CZWR-gen
CZWR-data
CZRD-addr
CZRhold
74a
80
Read Address decoder
84
CZRD-data
90
D
72
Write Address Decoder
we0
r0
82
MUX
decoder
add0
D
GEN
CON
92
we1
r1
r0
add1
74b
READ OUTPUT
General Registers
MUX
r1
Control Registers
we31
78
r31
86
add31
76
74c
r31
DSP

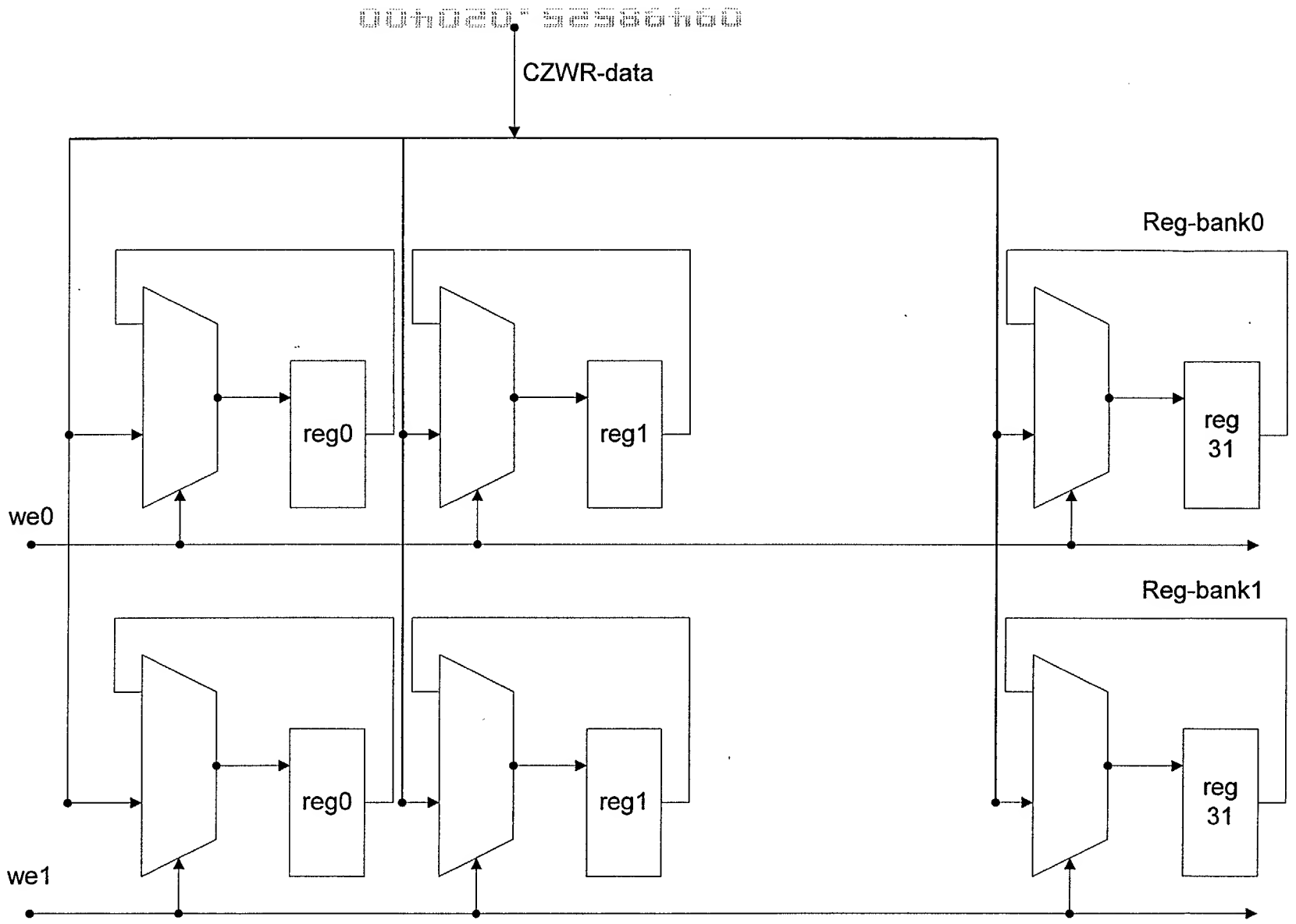

CZWR-data
Reg-bank0
reg0
reg1
reg 31
we0
Reg-bank1
reg0
reg1
reg 31
we1

FIG. 7B

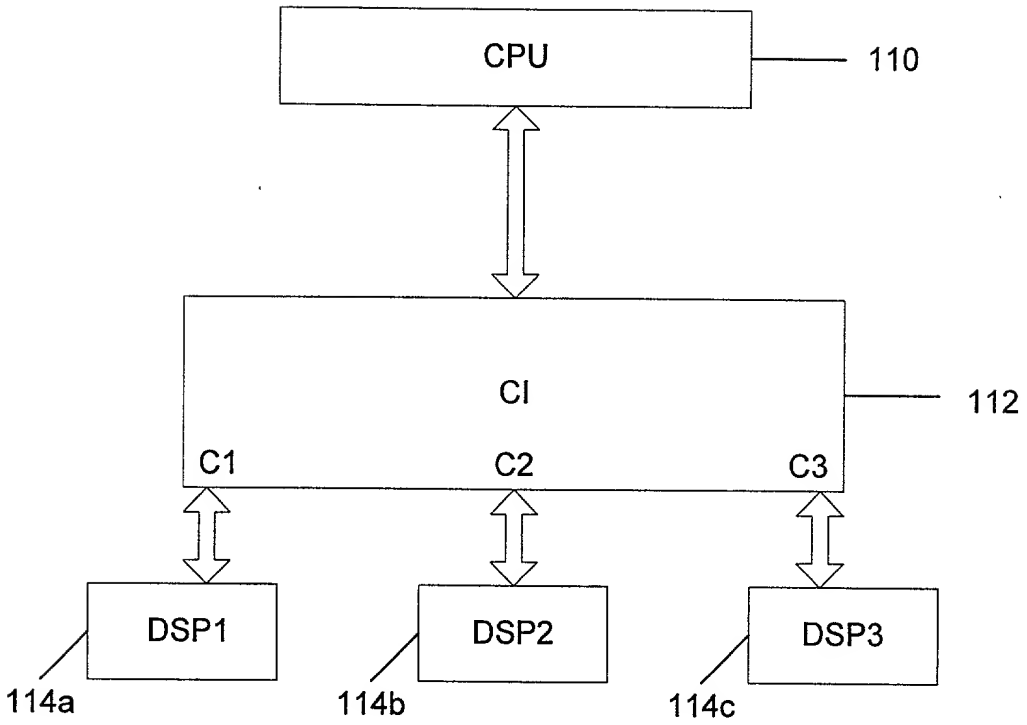

CPU
110
CI
112
C1
C2
C3
DSP1
DSP2
DSP3
114a
114b
114c

FIG. 8A

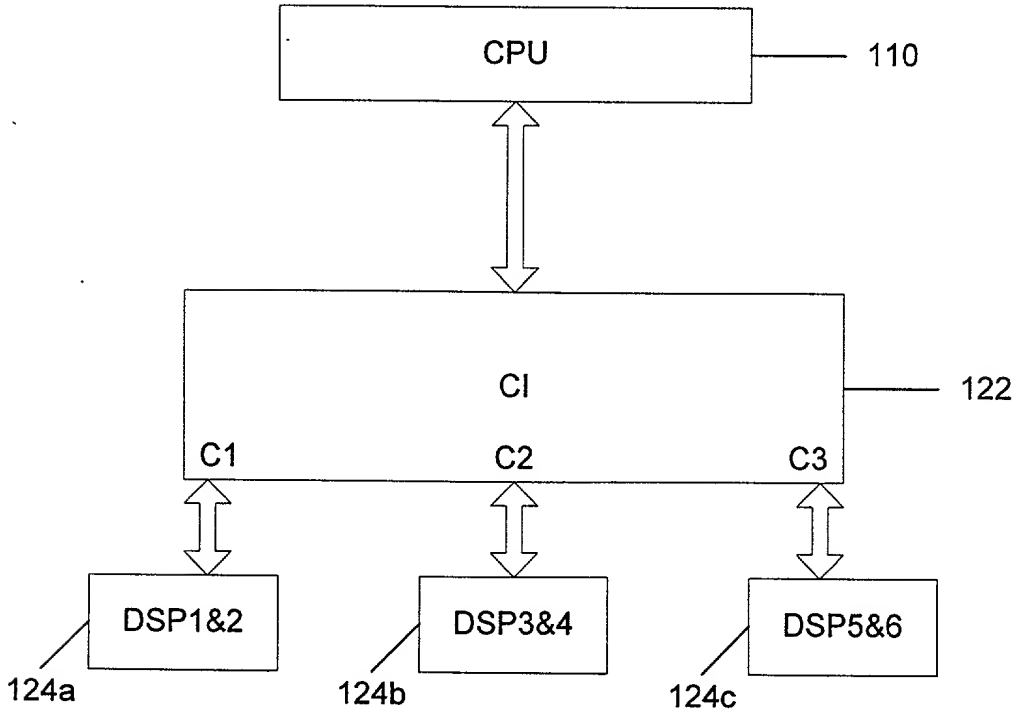

CPU
110
CI
122
C1
C2
C3
DSP1&2
DSP3&4
DSP5&6
124a
124b
124c

FIG. 8B

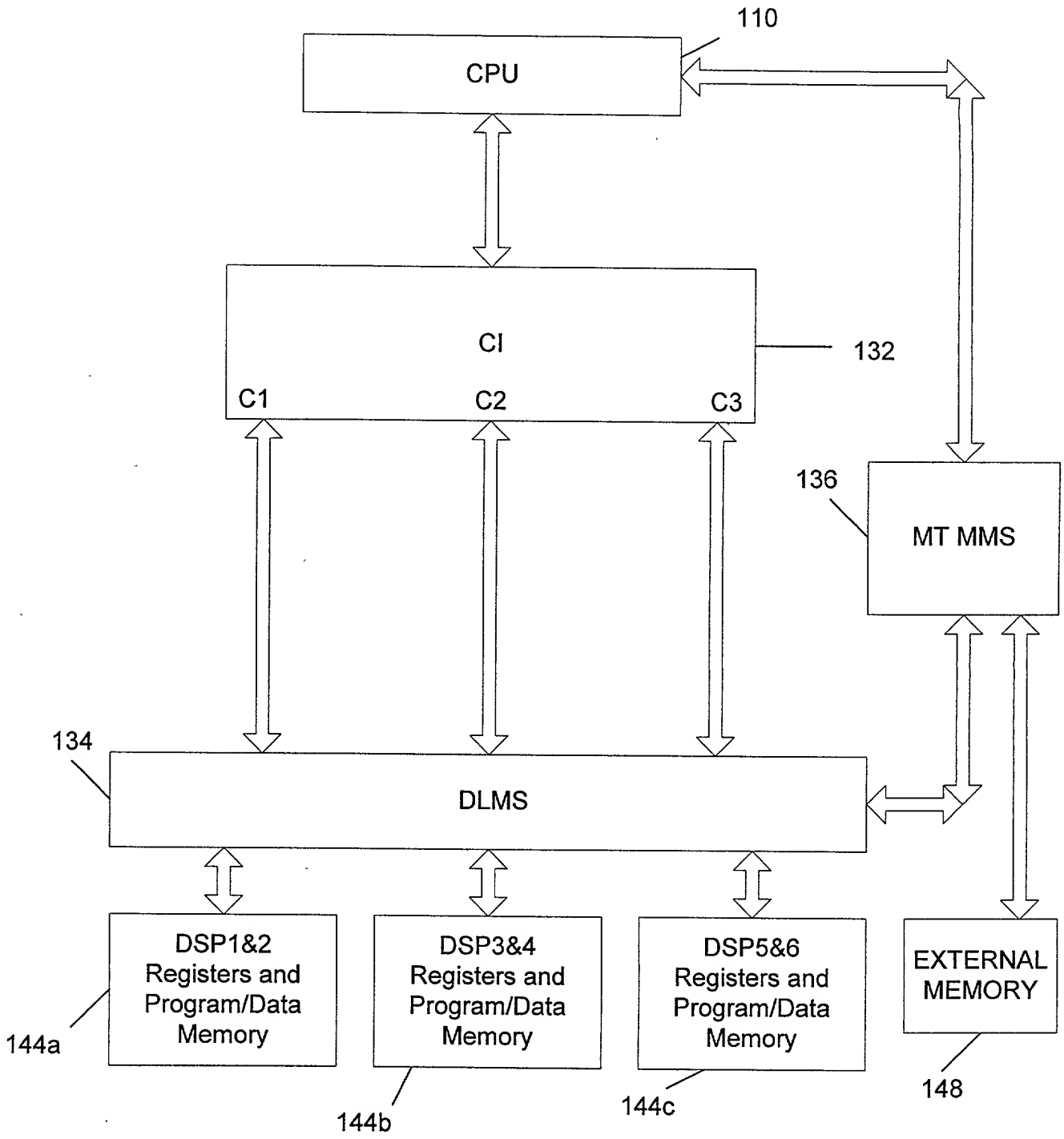

110
CPU
CI
132
C1
C2
C3
136
MT MMS
134
DLMS
DSP1&2 Registers and Program/Data Memory
DSP3&4 Registers and Program/Data Memory
DSP5&6 Registers and Program/Data Memory
EXTERNAL MEMORY
144a
144b
144c
148

FIG. 9A

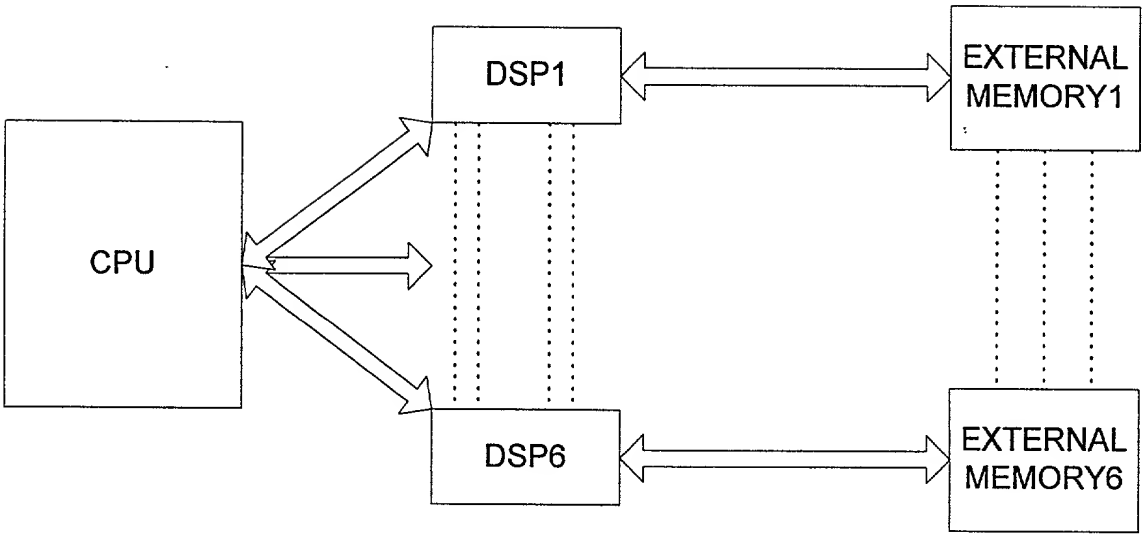


FIG. 9B

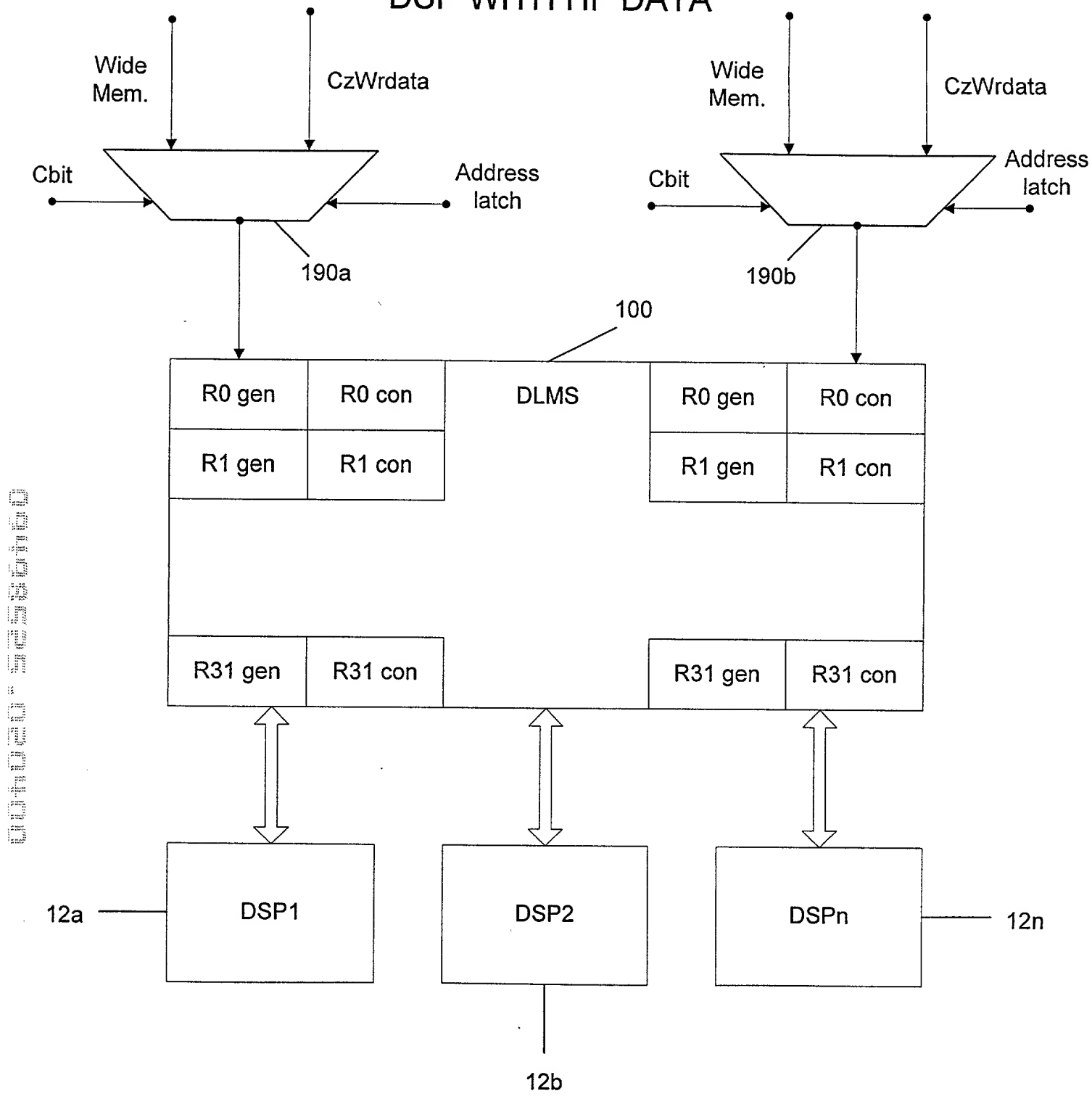

MT MMS LOADING
DSP WITH HF DATA
Wide Mem.
CzWrdata
Cbit
Address latch
190a
Wide Mem.
CzWrdata
Cbit
Address latch
190b
100
R0 gen
R0 con
DLMS
R0 gen
R0 con
R1 gen
R1 con
R1 gen
R1 con
R31 gen
R31 con
R31 gen
R31 con
12a
DSP1
DSP2
DSPn
12n
12b

FIG. 10

2 MAC CLUSTER

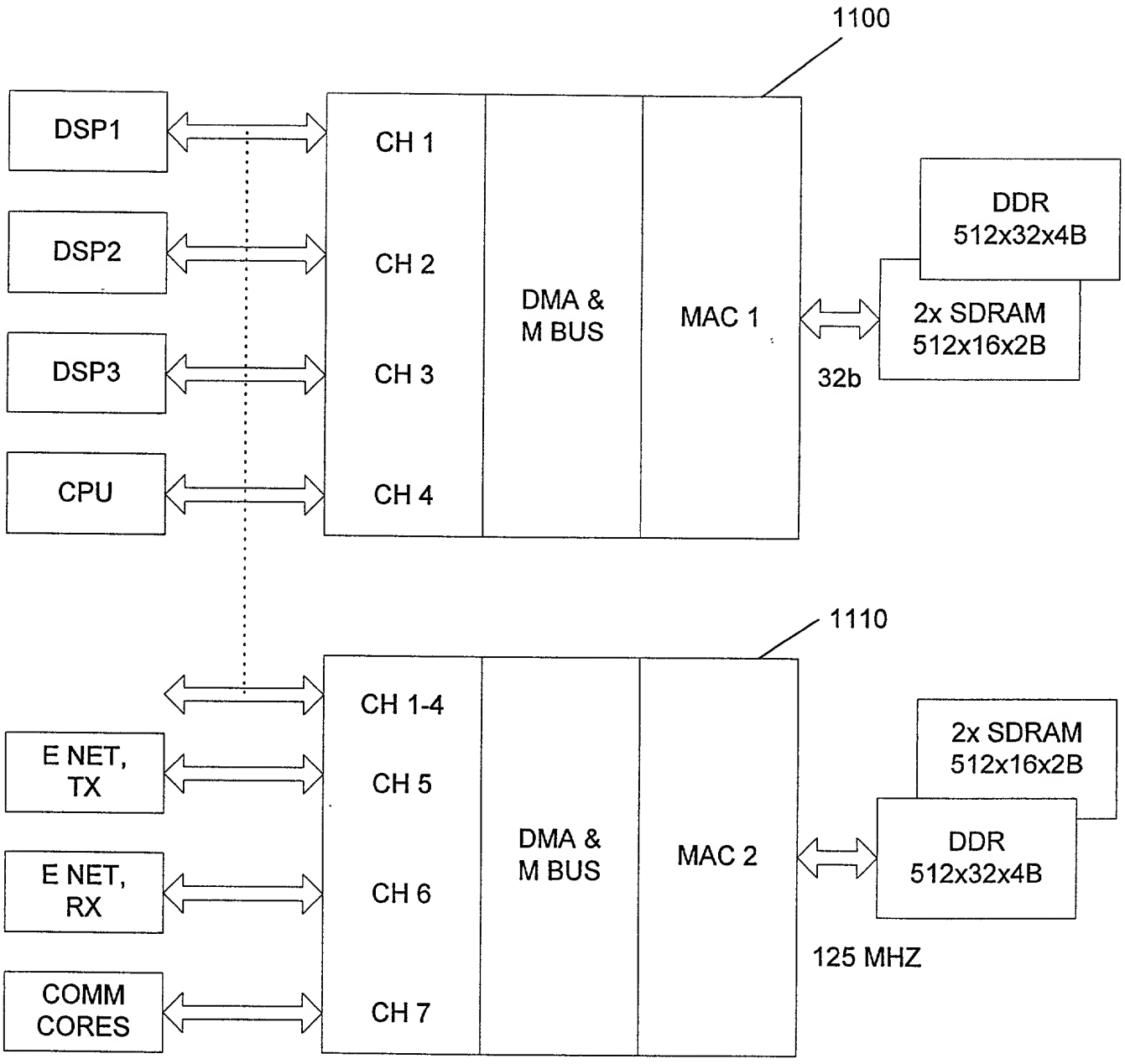


FIG. 11

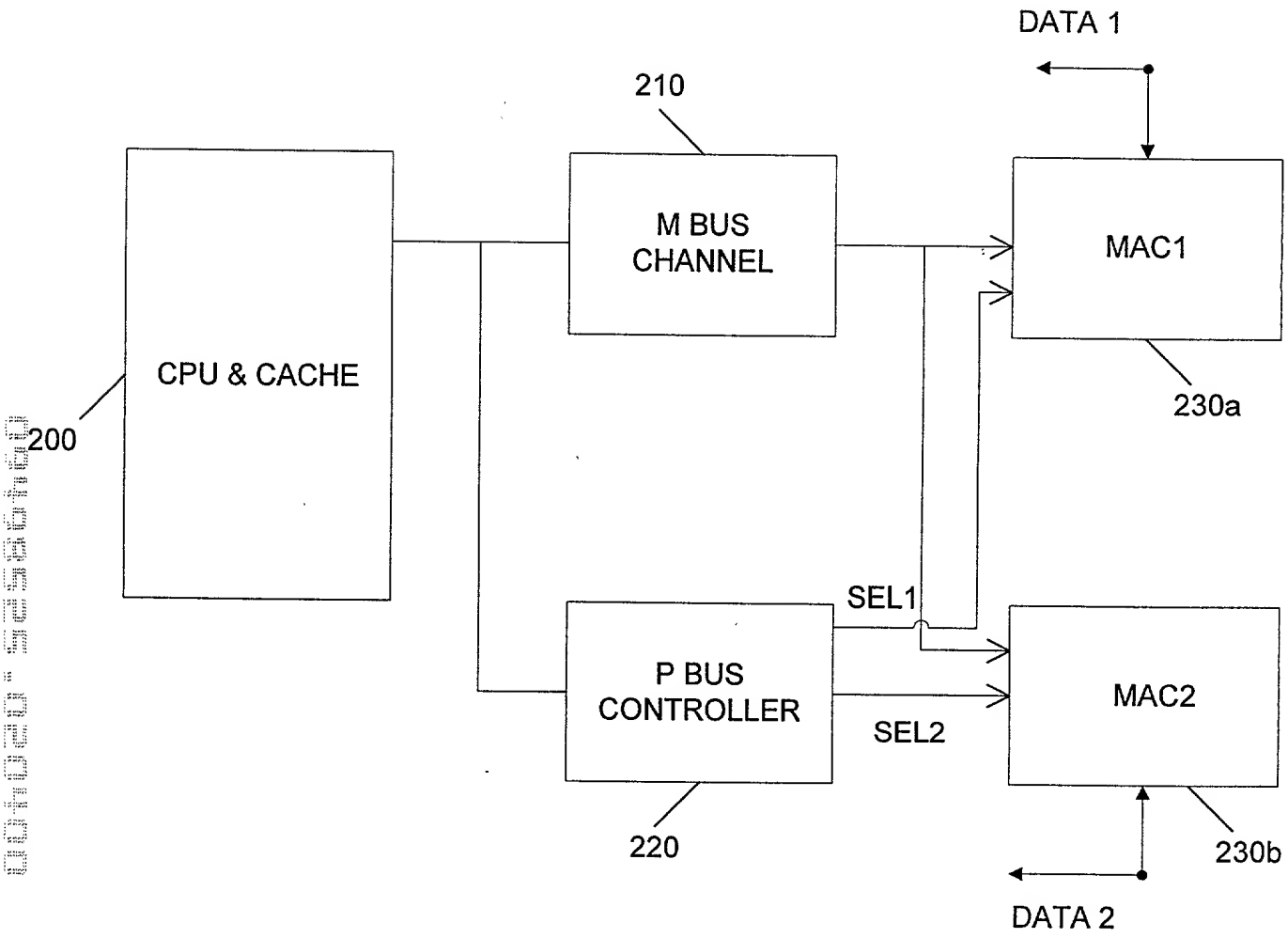

DATA 1
210
M BUS
CHANNEL
MAC1
CPU & CACHE
200
230a
SEL1
P BUS
CONTROLLER
MAC2
SEL2
220
230b
DATA 2

FIG. 12

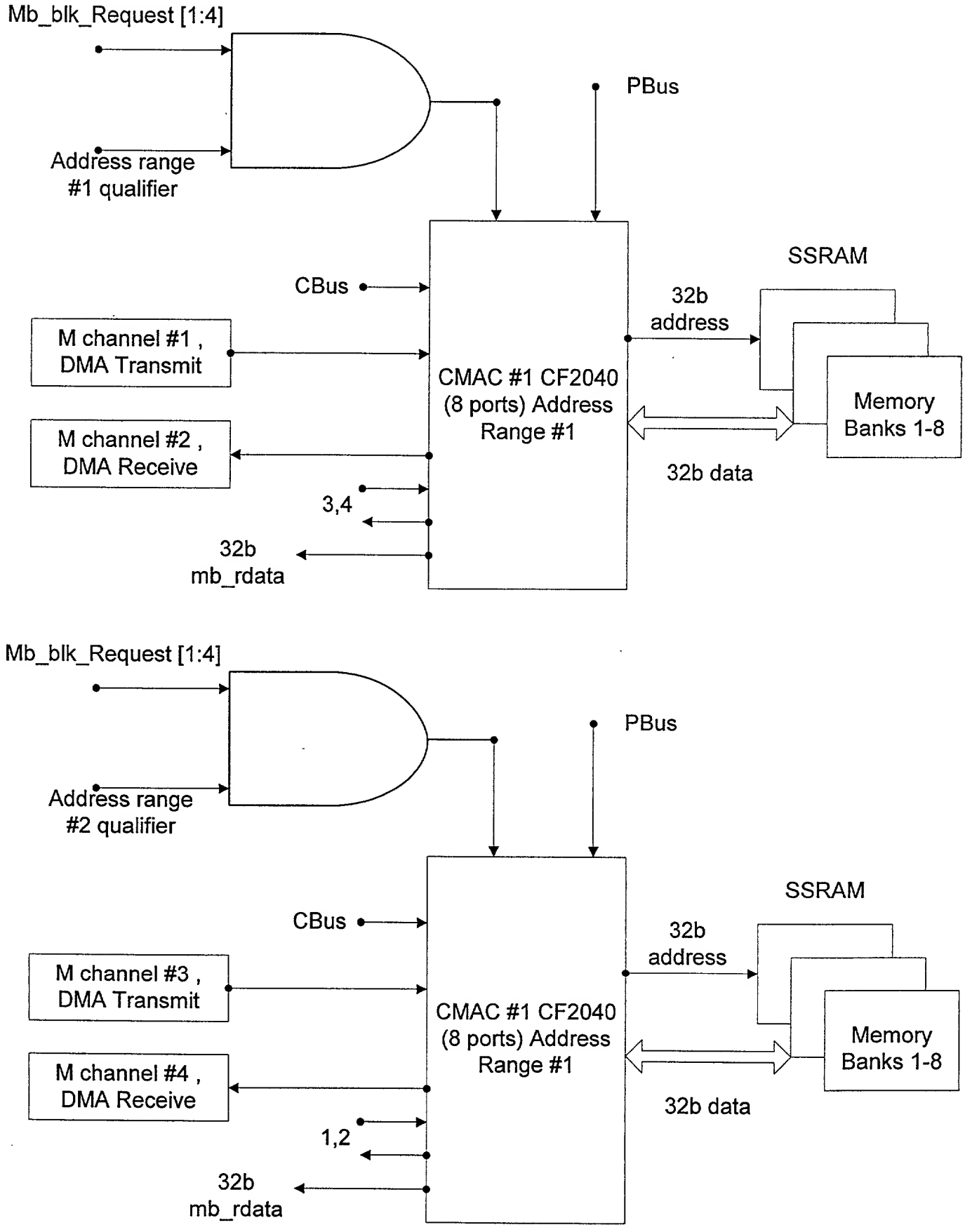

DMA & MAC CLUSTER
Mb_blk_Request [1:4]
Address range
#1 qualifier
PBus
SSRAM
CBus
32b
address
M channel #1 ,
DMA Transmit
CMAC #1 CF2040
(8 ports) Address
Range #1
Memory
Banks 1-8
M channel #2 ,
DMA Receive
32b data
3,4
32b
mb_rdata
Mb_blk_Request [1:4]
Address range
#2 qualifier
PBus
SSRAM
CBus
32b
address
M channel #3 ,
DMA Transmit
CMAC #1 CF2040
(8 ports) Address
Range #1
Memory
Banks 1-8
M channel #4 ,
DMA Receive
32b data
1,2
32b
mb_rdata

FIG. 13

MULTIPLEXING THE DATA BUS TO INDIVIDUAL CHANNELS

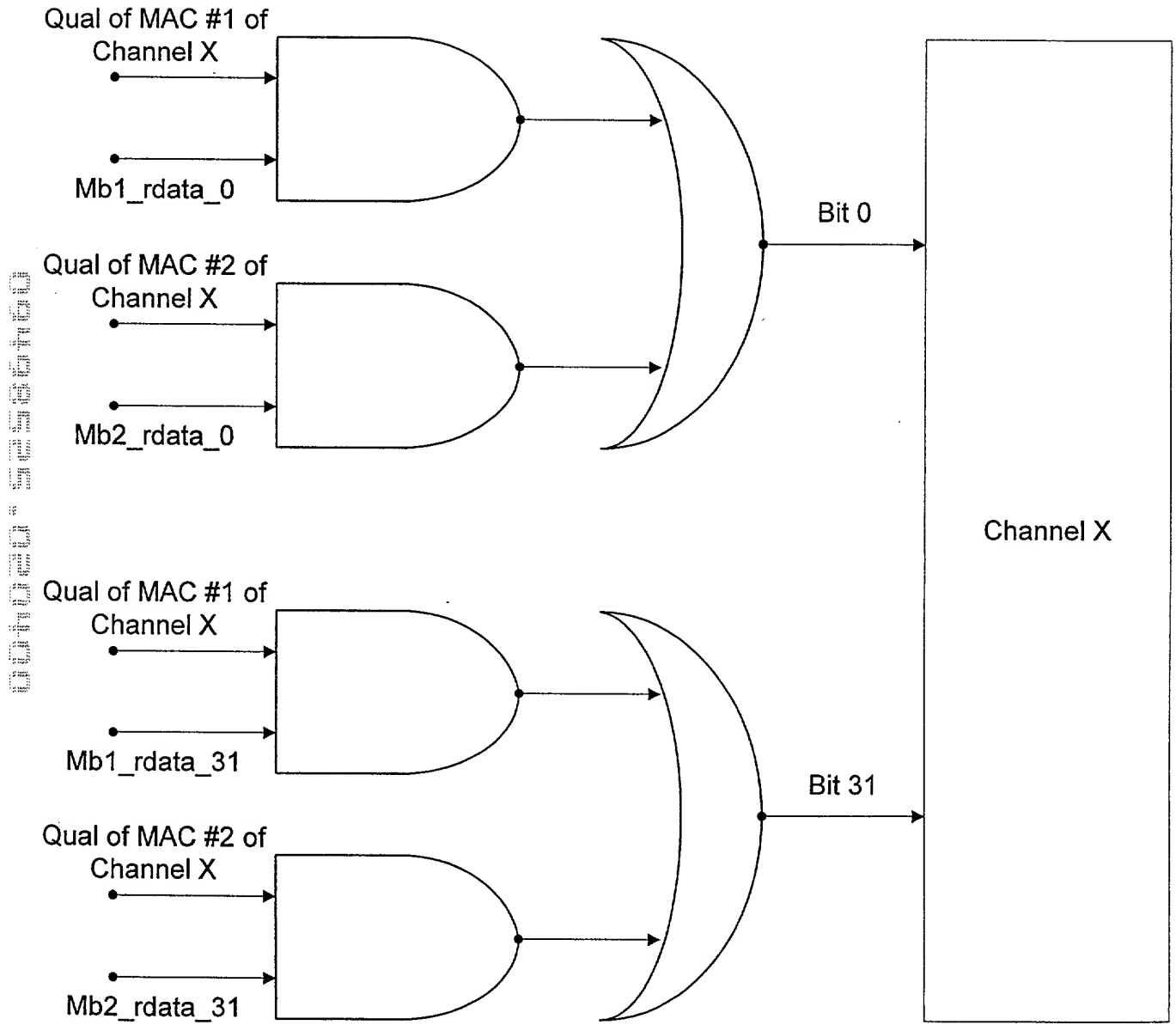
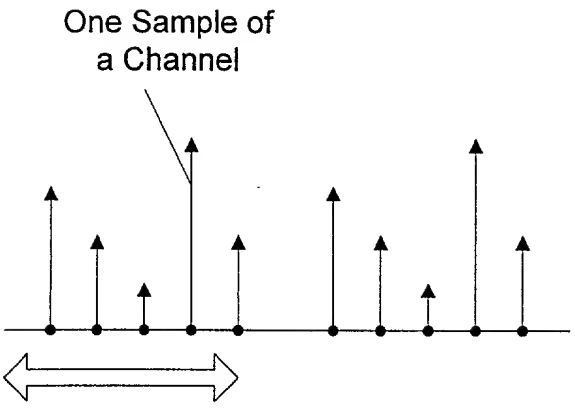


FIG. 14

REAL TIME LOAD MANAGEMENT



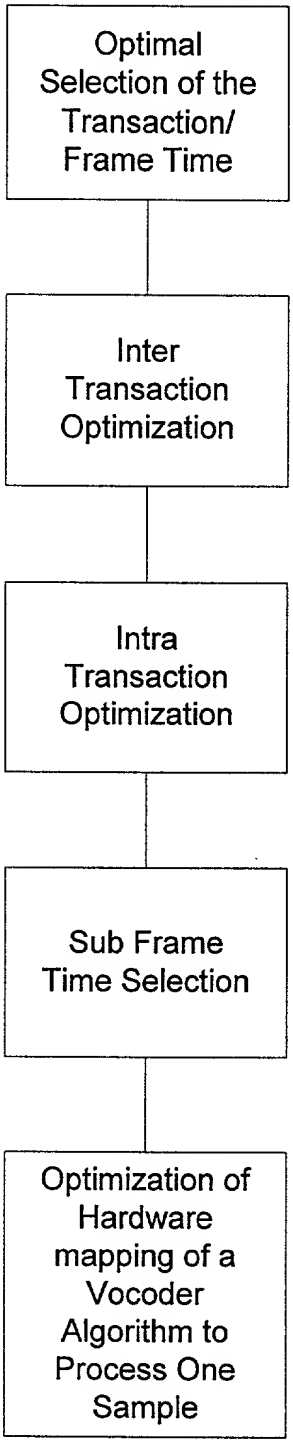


FIG. 15

DLMS PARALLEL WORD TRANSFER

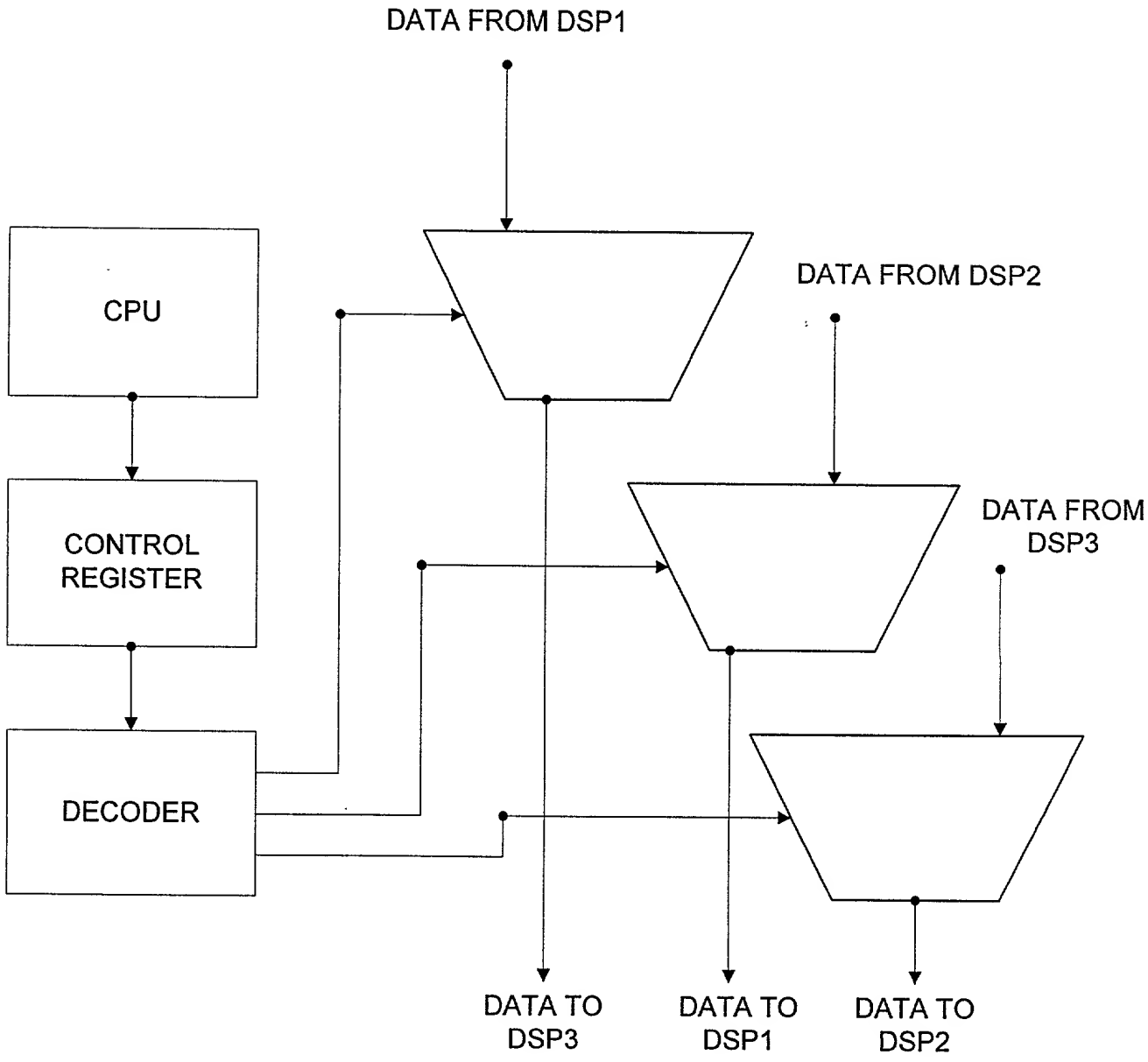


FIG. 16

PE ARRAY

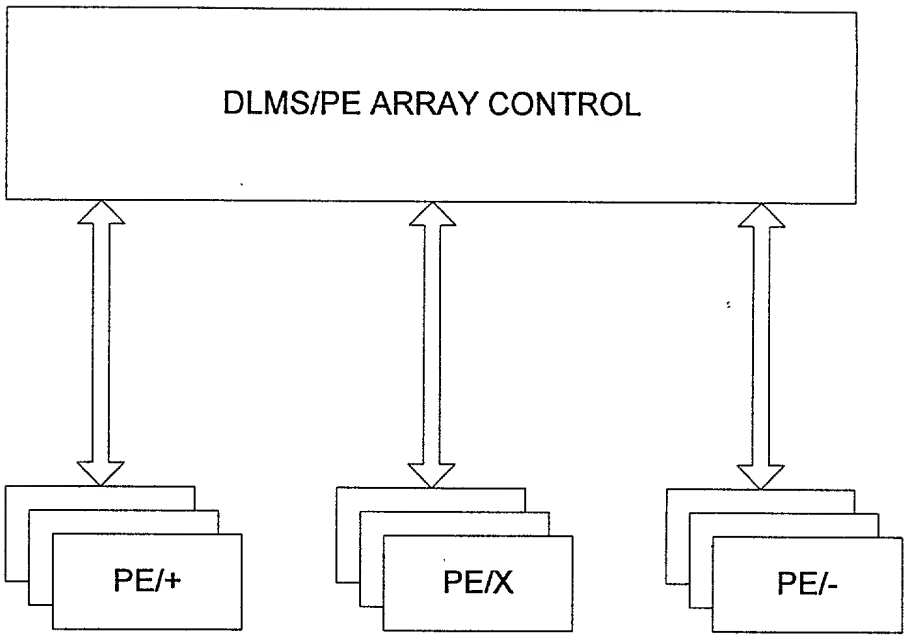


FIG. 17

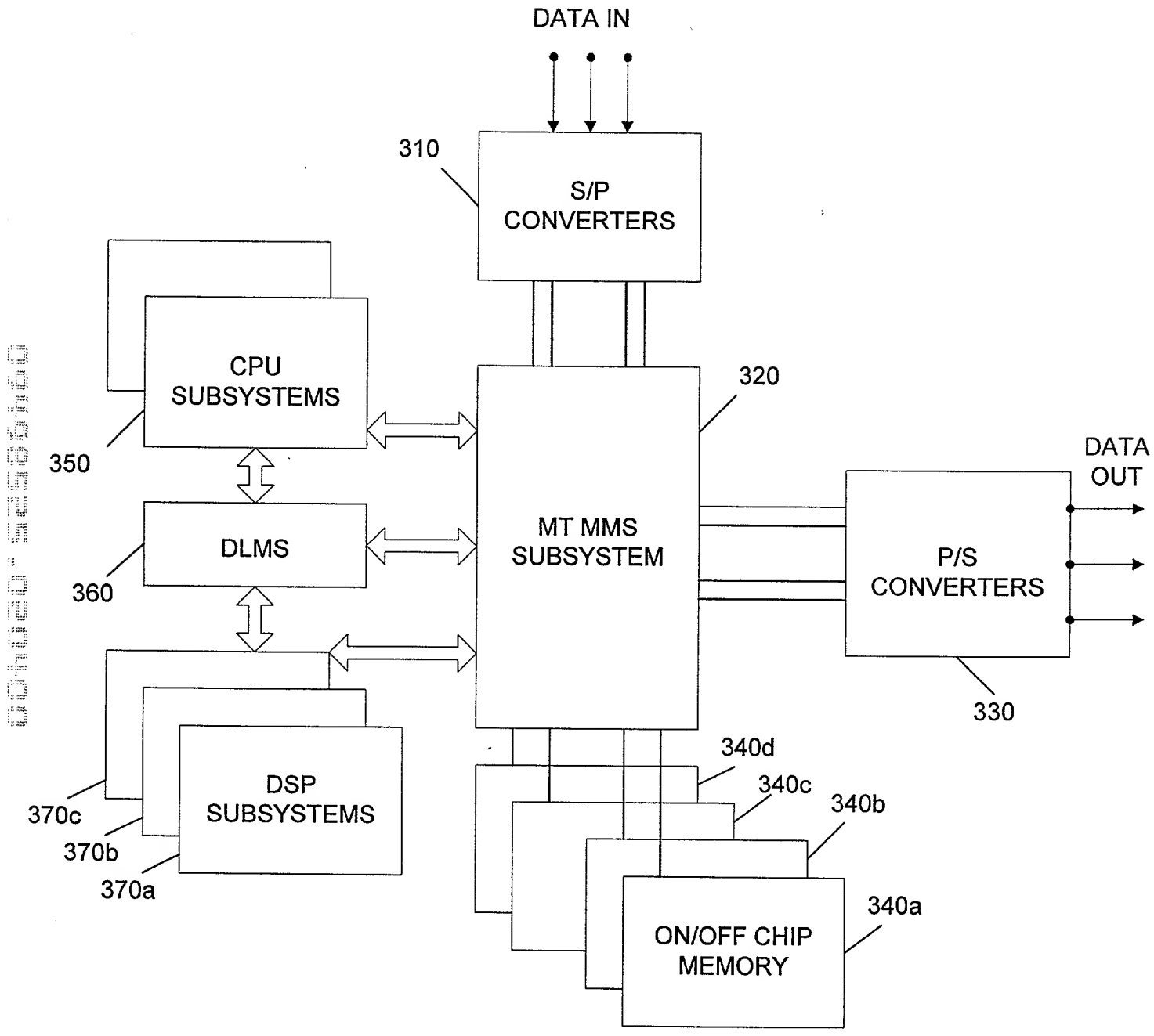

DATA IN
310
S/P
CONVERTERS
CPU
SUBSYSTEMS
350
320
DLMS
360
MT MMS
SUBSYSTEM
DATA
OUT
P/S
CONVERTERS
330
DSP
SUBSYSTEMS
370c
370b
370a
340d
340c
340b
340a
ON/OFF CHIP
MEMORY

FIG. 18A

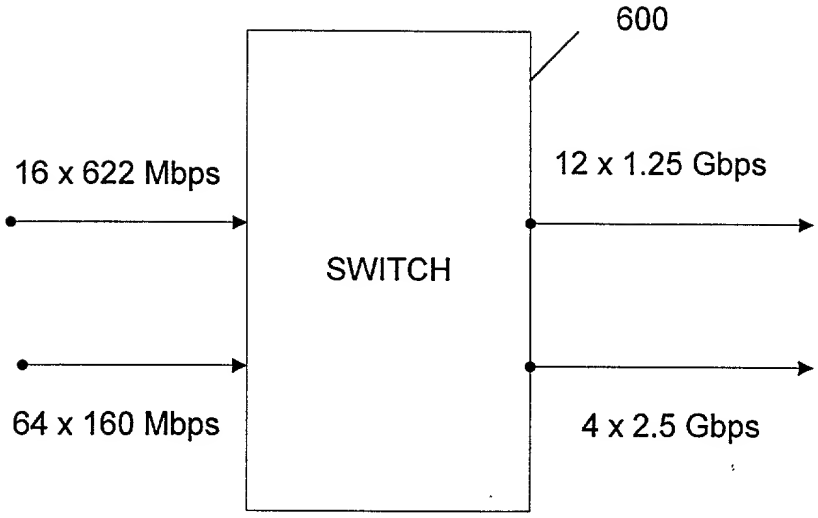


FIG. 18B

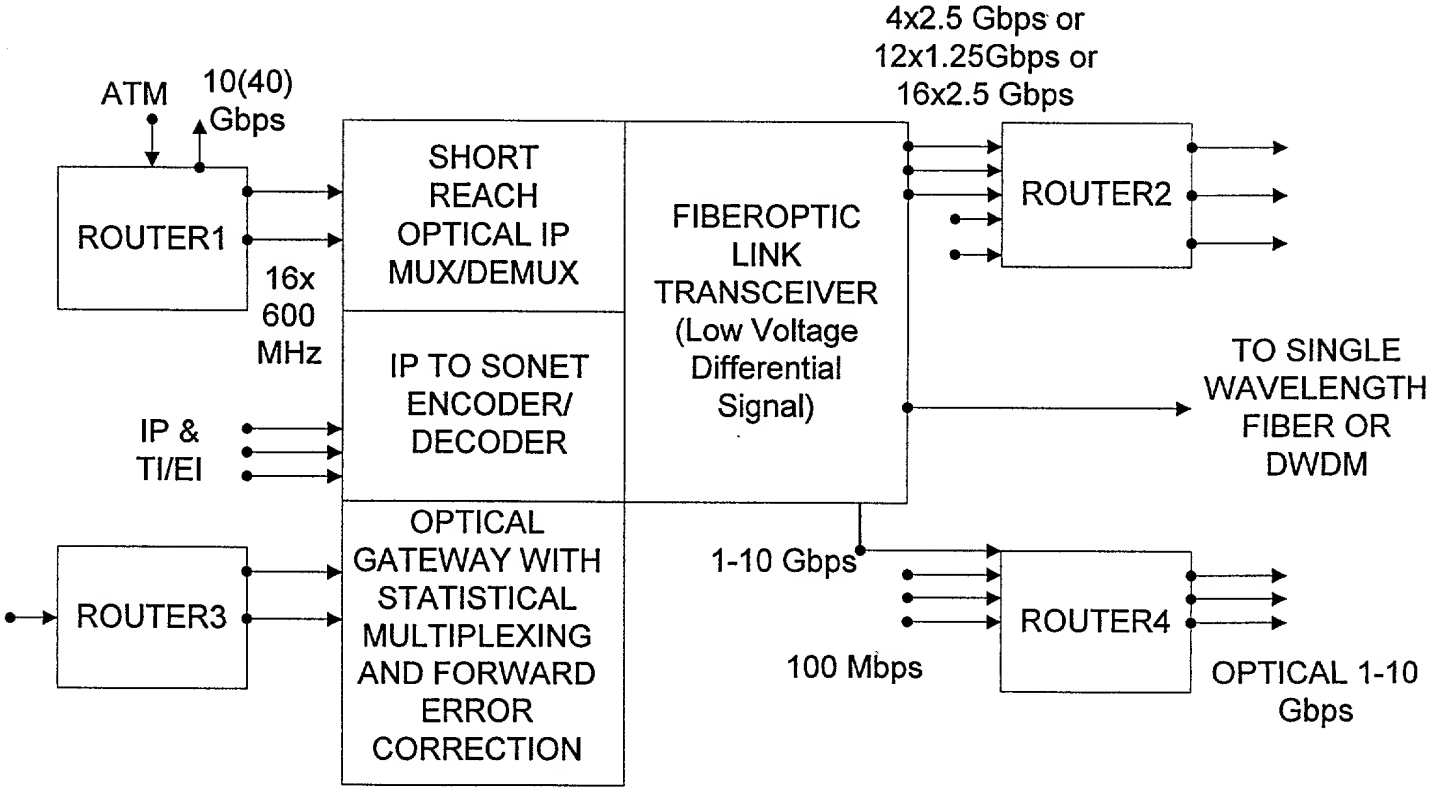


FIG. 18C

DECLARATION AND POWER OF ATTORNEY
FOR PATENT APPLICATION

As a below named inventor, I hereby declare that my residence, post office address and citizenship are as stated below next to my name, and I believe I am the original, first and sole inventor of the subject matter which is claimed and for which a patent is sought on the INVENTION ENTITLED **Real Time DSP Load Management System**, the specification of which is attached hereto, bearing Atty Docket No. 73234 / 0261856.

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above. I acknowledge the duty to disclose all information known to me to be material to patentability as defined in 37 C.F.R. 1.56. I hereby claim foreign priority benefits under 35 U.S.C. 119/365 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate filed by me or my assignee disclosing the subject matter claimed in this application and having a filing date (1) before that of the application on which priority is claimed, or (2) if no priority claimed, before the filing date of this application:

PRIOR FOREIGN APPLICATION(S): Number	Country	Day/MONTH/Year Filed	Date first Laid-open or Published:	Date Patented or Granted:	Priority Claimed? Yes ☐ No ☐

I hereby claim domestic priority benefit under 35 U.S.C. 119/120/365 of the indicated United States applications listed below and PCT international applications listed above or below and, if this is a continuation-in-part (CIP) application, insofar as the subject matter disclosed and claimed in this application is in addition to that disclosed in such prior applications, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in 37 C.F.R. 1.56 which became available between the filing date of each such prior application and the national or PCT international filing date of this application:

PRIOR U.S. PROVISIONAL, NONPROVISIONAL AND/OR PCT APPLICATIONS Application No.:	Day/MONTH/Year Filed:	Status (pending, abandoned, patented)	Priority Claimed? Yes ☐ No ☐

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

And I hereby appoint Pillsbury Madison & Sutro LLP, 1100 New York Avenue, N.W., Ninth Floor, East Tower, Washington, D.C. 20005-3918, tel. (650) 233-4790 (to whom all communications are to be directed), and the below-named persons (of the same address) individually and collectively my attorneys to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith and with the resulting patent, and I hereby authorize them to delete names of persons no longer with their firm and to act and rely on instructions from and communicate directly with the assignee which first sent this case to them and by which I hereby declare that I have consented after full disclosure to be represented, unless/until I instruct the above Firm in writing to the contrary.

Paul N. Kokulis	16773	Dale S. Lazar	28872	Timothy J. Klima	34852	Michael R. Dzwonczyk	36787
Raymond F. Lippitt	17519	Glenn J. Perry	28458	Stephen C. Glazier	31361	W. Patrick Bengtsson	32456
G. Lloyd Knight	17698	Kendrew H. Colton	30368	Paul F. McQuade	31542	Jack S. Barufka	37087
Carl G. Love	18781	Paul E. White, Jr.	32011	Ruth N. Mordueh	31044	Adam R. Hess	41835
Kevin E. Joyce	20508	G. Paul Edgell	24238	Richard H. Zaitlen	27248		
George M. Sirilla	18221	Lynn E. Eccleston	35861	Roger R. Wise	31204		
Donald J. Bird	25323	David A. Jakopin	32995	Jay M. Finkelstein	21082		
Peter W. Gowdey	25872	Mark G. Paulson	30793	Anita M. Kirkpatrick	32617		

INVENTOR'S SIGNATURE: Hammam Elabd Date 2/3/00

Inventor's Name: **ELABD, Hammam** Country of Citizenship: **United States of America**
Residence (City, State): **Sunnyvale, California**
Post Office Address: **587 Middlebury Drive**
Sunnyvale, CA 94087

Rule 56(a) & (b) = 37 C.F.R. 1.56(a) & (b)
PATENT AND TRADEMARK CASES - RULES OF PRACTICE
DUTY OF DISCLOSURE

(a) ... Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the [Patent and Trademark] Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability...(b) information is material to patentability when it is not cumulative and (1) It also establishes by itself, or in combination with other information, a prima facie case of unpatentability of a claim or (2) refers, or is inconsistent with, a position the applicant takes in: (i) Opposing an argument of unpatentability relied on by the Office, or (ii) Asserting an argument of patentability.

PATENT LAWS 35 U.S.C.

§102. Conditions for patentability; novelty and loss of right to patent

A person shall be entitled to a patent unless--

(a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for patent or

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of the application for patent in the United States, or

(c) he has abandoned the invention, or

(d) the invention was first patented or caused to be patented, or was the subject of an inventor's certificate, by the applicant or his legal representatives or assigns in a foreign country prior to the date of the application for patent in this country on an application for patent or inventor's certificate filed more than twelve months* before the filing of the application in the United States, or

(e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent, or

(f) he did not himself invent the subject matter sought to be patented, or

(g) before the applicant's invention thereof the invention was made in this country by another who had not abandoned, suppressed, or concealed it. In determining priority of invention there shall be considered not only the respective dates of conception and reduction to practice of the invention, but also the reasonable diligence of one who was first to conceive and last to reduce to practice, from a time prior to conception by the other.

§103. Condition for patentability; non-obvious subject matter

A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made. Subject matter developed by another person, which qualified as prior art only under subsection (f) or (g) of section 102 of this title, shall not preclude patentability under this section where the subject matter and the claimed invention were, at the time the invention was made, owned by the same person or subject to an obligation of assignment to the same person.

* Six months for Design Applications (35 U.S.C. 172).